VŠB – Technická univerzita Ostrava
Fakulta strojní
Katedra automatizační techniky a řízení

Hardwarový simulátor technologických procesů pro řídicí systém ABB/AC800

Hardware Simulator of Technological Processes for ABB/AC800 Control System

Student: Bc. Renáta Havránková
Vedoucí diplomové práce: doc. Ing. Lenka Landryová, CSc.

Ostrava 2015

VŠB - Technická univerzita Ostrava
Fakulta strojní
Katedra automatizační techniky a řízení

Zadání diplomové práce

Student: **Bc. Renáta Havránková**

Studijní program: N2301 Strojní inženýrství

Studijní obor: 3902T004 Automatické řízení a inženýrská informatika

Téma: Hardwarový simulátor technologických procesů pro řídicí systém ABB/AC800
Hardware Simulator of Technological Processes for ABB/AC800 Control System

Zásady pro vypracování:

1. Zjistěte možnosti využití různých programovacích nástrojů v prostředí ABB 800xA (Control Builder, Function Designer, Graphic Builder, Panel Builder) k řešení jedné úlohy a svá zjištění zdokumentujte.
2. Popište hw simulátor technologického procesu. Otestujte komunikaci mezi hw simulátorem a kontrolérem ABB řady AC800, vstupními a výstupními (I/O) kartami a operátorským panelem ABB PD 835.
3. Prostudujte funkční popisy stávajících knihoven využívaných v projektech nasazených aplikací firmy ABB.
4. Pro HW simulátor vytvořte základní knihovnu, navrhněte a vytvořte kód pro funkční bloky pro zadávání a zobrazení zadaných hodnot a alarmových limitů.
5. Navrhněte a vytvořte grafiku pro vizualizaci HW simulátoru na operátorském pracovišti.
6. Ověřte funkčnost hardware simulátoru, zhodnoťte dosažené výsledky a navrhněte směr dalšího řešení návrhu úlohy.

Seznam doporučené odborné literatury:

Zavadil, J. Systém monitorování a řízení technologie. Ostrava:
katedra automatizační techniky a řízení, VŠB-TU Ostrava, 2009. 57 stran.
Diplomová práce, vedoucí: Landryová, L.
ABB, S800 I/O Modules and Termination Units, System Version 5.1, 2013. 668 stran.
ABB, AC 800M Controller Hardware, System Version 5.1, 2013.452 stran.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Lenka Landryová, CSc.**

Datum zadání: 13.12.2014
Datum odevzdání: 18.05.2015

VYSOKÁ ŠKOLA BÁŇSKÁ - TECHNICKÁ UNIVERZITA Fakulta strojní 3 OSTRAVA

doc. Ing. Renata Wagnerová, Ph.D.
vedoucí katedry

doc. Ing. Ivo Hlavatý, Ph.D.
děkan fakulty

Místopřísežné prohlášení studenta

Prohlašuji, že jsem celou diplomovou práci včetně příloh vypracovala samostatně pod vedením vedoucího diplomové práce a uvedla jsem všechny použité podklady a literaturu.

V Ostravě 18.5.2015 ..... Zuzana Navrátilová .....

Podpis studenta

Prohlašuji, že

- jsem byla seznámena s tím, že na moji diplomovou práci se plně vztahuje zákon č.121/2000 Sb., autorský zákon, zejména § 35 - užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a § 60 - školní dílo.

- beru na vědomí, že Vysoká škola báňská - Technická univerzita Ostrava (dále jen „VŠB-TUO") má právo nevýdělečně ke své vnitřní potřebě diplomovou práci užít (§ 35 odst. 3).

- souhlasím s tím, že diplomová práce bude v elektronické podobě uložena v Ústřední knihovně VŠB-TUO k nahlédnutí a jeden výtisk bude uložen u vedoucího diplomové práce. Souhlasím s tím, že údaje o kvalifikační práci budou zveřejněny v informačním systému VŠB-TUO.

- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona.

- bylo sjednáno, že užít své dílo-diplomovou práci nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).

- beru na vědomí, že odevzdáním své práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, bez ohledu na výsledek její obhajoby.

V Ostravě: 18.5.2015

Renáta Havrlánková

Podpis

Jméno a příjmení autora práce: RENÁTA HAVRLÁNKOVÁ

Adresa trvalého pobytu autora práce: A.G.L. SVOBODY 752
STUDÉNKA 2
742 13

Poděkování

Chtěla bych poděkovat své vedoucí diplomové práce doc. Ing. Lence Landryové, CSc. za odborné vedení, za pomoc a rady při zpracování této práce. Dále bych chtěla poděkovat Ing. Lukáši Hédlovi za odbornou pomoc, ochotu a trpělivost, které mi v průběhu zpracování diplomové práce věnoval. Mé poděkování také patří mé rodině za jejich pomoc a podporu během studia.

ANOTACE DIPLOMOVÉ PRÁCE

HAVRÁNKOVÁ, R., *Hardwarový simulátor technologických procesů pro řídicí systém ABB AC800: diplomová práce.* Ostrava: VŠB – Technická univerzita Ostrava, Fakulta strojní, Katedra automatizační techniky a řízení, 2015, 63 s. Vedoucí práce: Landryová, L.

Diplomová práce se zabývá návrhem knihoven pro řídicí systém ABB AC800. V teoretické části je popsáno programovací prostředí, ve kterém je projekt vypracováván, je zde popis reálného hardwaru a vysvětlení knihoven a jejich funkčních popisů. Praktická část diplomové práce obsahuje návrh knihovny se základními prvky potřebnými pro ovládání simulátoru, vytvoření těchto modulů, aplikace a vizualizace technologického procesu napouštění a vypouštění nádrže a ochlazování a ohřívání teploty v nádrži. Dále je zajištěna komunikace mezi hardwarem a grafickým displejem na PC a komunikace hardwaru a operátorského panelu PP835. Cílem diplomové práce je vytvořit základní knihovnu, která následně bude používána jako školící a demonstrační materiál pro firmu ABB s.r.o.

ANOTATITON OF MASTER THESIS

HAVRÁNKOVÁ, R. *Hardware Simulator of Technological Processes for ABB/AC800 Control System: Master Thesis.* Ostrava: VŠB – Technical University of Ostrava, Faculty of Mechanical Engineering, Department of Control Systems and Instrumentation, 2015, 63 s. Thesis head: Landryová, L.

This thesis describes the design of libraries for control system ABB AC 800M. The theoretical part describes the programming space in which the project is drawn up, there is a real hardware description and explanation of libraries and their functional descriptions. The practical part contains the design of library with the basic elements needed to controlling a simulator, creation of these modules, applications and visualization of technological process of filling and emptying of the tank and the cooling and heating temperature in the tank. Next is ensured communication between the hardware and the graphic display on the PC and communications between hardware and operator panel PP835. The aim of the thesis is to create a basic library, which will subsequently be used as material for a training and demonstration material for the company ABB s.r.o.

Obsah

Seznam použitých značek a symbolů

AIS – analogový vstupní signál (z angl. Analog Input Signal)

AOS – analogový výstupní signál (z angl. Analog Output Signal)

Apod. – a podobně

Atd. – a tak dále

CM – řídicí modul (z angl. Control Module)

DC – stejnosměrný proud

DIS – digitální vstupní signál (z angl. Digital Input Signal)

DOS – digitální výstupní signál (z angl. Digital Output Signal)

HMI – rozhraní mezi člověkem a strojem (z angl. Human Machine Interface)

HW – hardware

IP – internetový protokol

I/O – vstupně/výstupní (z angl. Input/Output)

MMS – protokol k přenosu dat (z angl. Microsoft Media Server)

OPC – propojování a vkládání objektů pro řízení procesů

Úvod

Lidé se od nepaměti snaží ulehčit si práci technologiemi. Začalo to vyráběním jednoduchých seker až po vyrábění obrovských těžkých strojů. S mechanizací technologických procesů jde ruku v ruce automatizace těchto procesů. Mechanizací si lidé pomáhají při práci, kdežto automatizací tuto práci přenechávají strojům. Automatizace umožňuje sledování a řízení stroje jako i celého technologického procesu. Díky automatizaci můžeme efektivněji a rychleji sledovat práci strojů a zabránit jejich poškození, popřípadě havárii celého procesu a vysokým škodám ekologickým i finančním.

Ve světě je mnoho firem zabývajících se automatizací. Jednou z nich je firma ABB s.r.o. Tato firma působí v několika odvětvích, jako je procesní automatizace, automatizace pohonů, atd. Firma ABB umožňuje studentům škol podílet se na automatizaci a psát diplomové práce v jejich firmě prostřednictvím programu ABB University. Díky tomuto programu jsem se do firmy ABB dostala i já a zpracovala tuto diplomovou práci.

Cílem této diplomové práce je navrhnout knihovnu pro řídicí systém ABB/AC800, která bude použita pro demonstrační a studijní účely v rámci firmy ABB. Firma má již spoustu úloh, ale tato diplomová práce bude vytvořena přímo na míru technickým prostředkům školící laboratoře. Její využití bude různé, například pro usnadnění přístupu studentům k projektu, jako školící prostředek ABB University a pro školení nových pracovníků firmy, popřípadě jako ukázka pro zákazníky firmy.

V této práci nejprve popíši programovací nástroje používané v prostředí ABB 800xA, dále popíšu hardware použitý ve školící laboratoři, vyzkouším komunikaci mezi HW simulátorem a kontrolérem ABB AC800M a poté i komunikaci HW simulátoru s panelem ABB PP835. Dále popíši podstatu vytváření knihoven a funkční popis stávajících knihoven použitých v projektech firmy ABB. Ve druhé části práce se budu zabývat vytvořením základní knihovny a funkčních bloků pro zobrazování zadaných hodnot a alarmových limitů, vytvářením vizualizace HW simulátoru na operátorském pracovišti pomocí grafického displeje a pomocí operátorského panelu ABB PP835. Nakonec ověřím funkčnost hardwarového simulátoru a zhodnotím dosažené výsledky.

1 Programovací nástroje v prostředí ABB 800xA

Systém 800xA je komplexní systém pro automatizaci. Jedná se o prostředí obsahující souhrn programů k vytváření softwaru, grafiky a pro management alarmů a trendů. Jde o bezpečné prostředí, protože má i funkce pro ověřování přístupu uživatelů. Výhoda tohoto prostředí spočívá v tom, že systém jedním kliknutím operátora získá potřebné informace o daném procesu, případně modulu nezávisle na tom, kde se operátor nachází. Pro vytváření projektu je využíváno mnoho programovacích nástrojů a prostředí. Vytvořený projekt je následně nahrán do kontroléru AC800M, propojení s počítačem je provedeno přes řídicí síť (Control Network). Řídicí síť je privátní síť s doménou pro data reálného času i pro všeobecný komunikační systém mezi několika průmyslovými počítači.

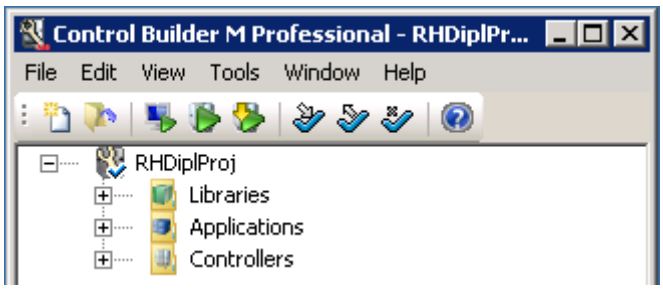
K vytvoření softwaru je nutno použit několik programovacích nástrojů a prostředí:

- Control Builder M Professional
- Function Designer
- Graphic Builder
- Panel Builder 800
- OPC Server 5.1.1
- SoftController 5.1.1

Nyní popíšu jednotlivé programy, které jsem při vytváření projektu použila.

1.1 Programovací prostředí Control Builder M Professional

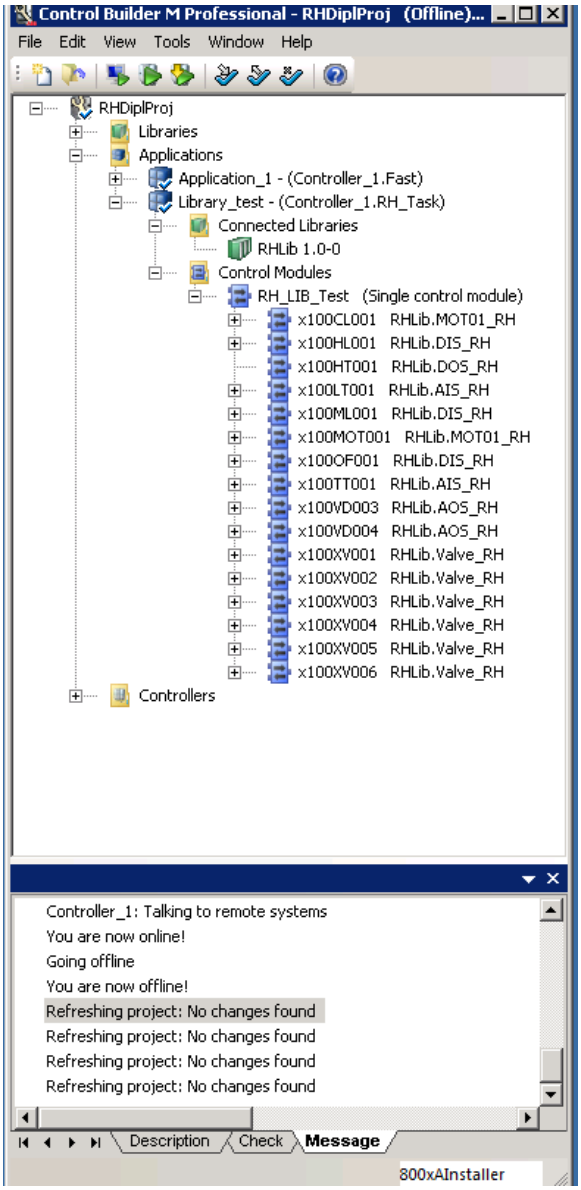
Hlavním programovacím nástrojem při vytváření projektu je Control Builder M Professional. V tomto prostředí vidíme projekt ve stromové struktuře.



Obrázek 1 Stromová struktura v programu Control Builder

Tento strom obsahuje 3 hlavní složky. Složka Knihovny (Libraries) obsahuje všechny knihovny, které jsou v rámci projektu použity. Tématu knihoven se budu ještě blíže věnovat v kapitolách 3 a 4.

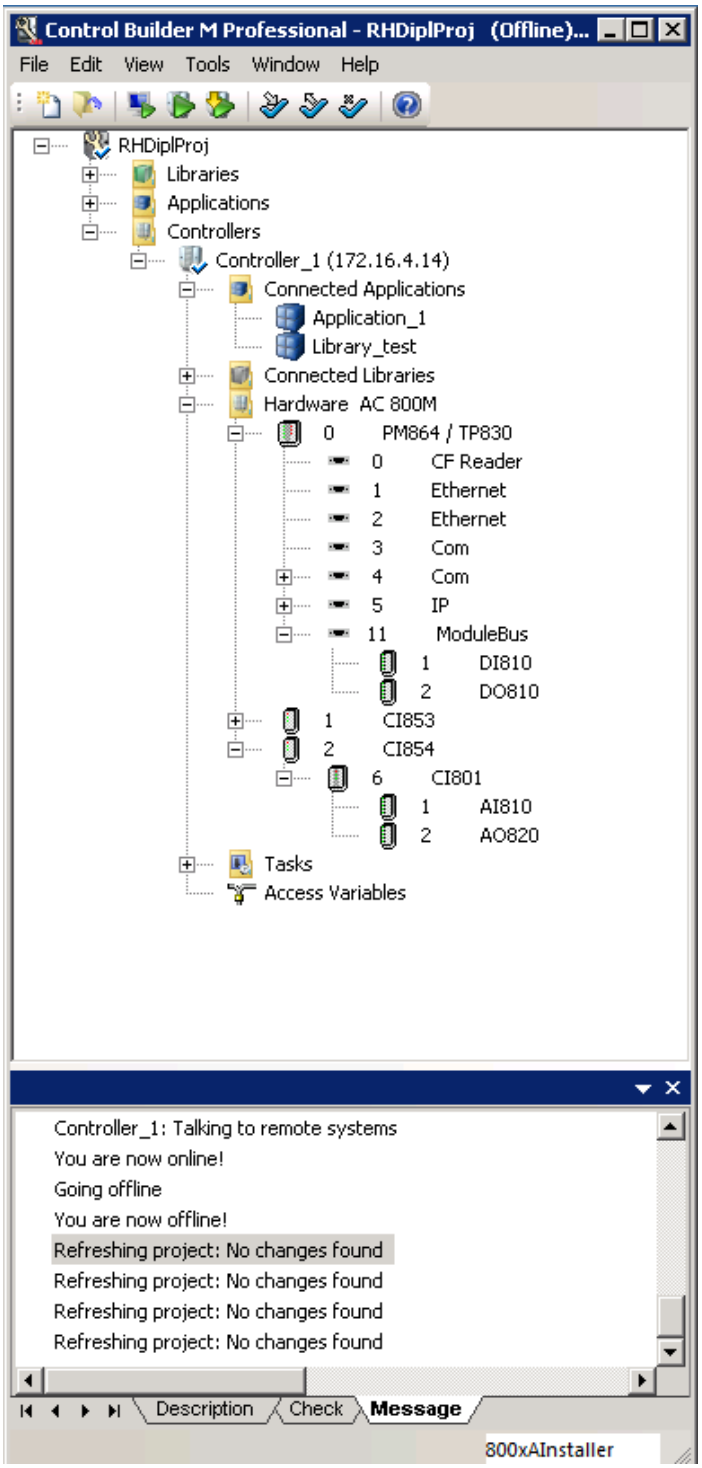
Druhá složka Aplikace (Applications) obsahuje námi vytvořené aplikace pro daný projekt. Takových aplikací může být několik. Složka Aplikace (Applications) je dále složena ze tří podsložek: složka Připojené knihovny (Connected Libraries), kde se nacházejí knihovny použité v aplikaci, dále složka Kontrolní moduly (Control Modules), kde se vytváří logika projektu a poslední složka Diagramy (Diagrams), kde můžeme programovat pomocí grafických programovacích jazyků (Obrázek 2).



Obrázek 2 Stromová struktura složky Aplikace

Třetí složka Kontroléry (Controllers) obsahuje seznam kontrolérů a každý z těchto kontrolérů obsahuje: připojené aplikace (Connected Applications), připojené knihovny

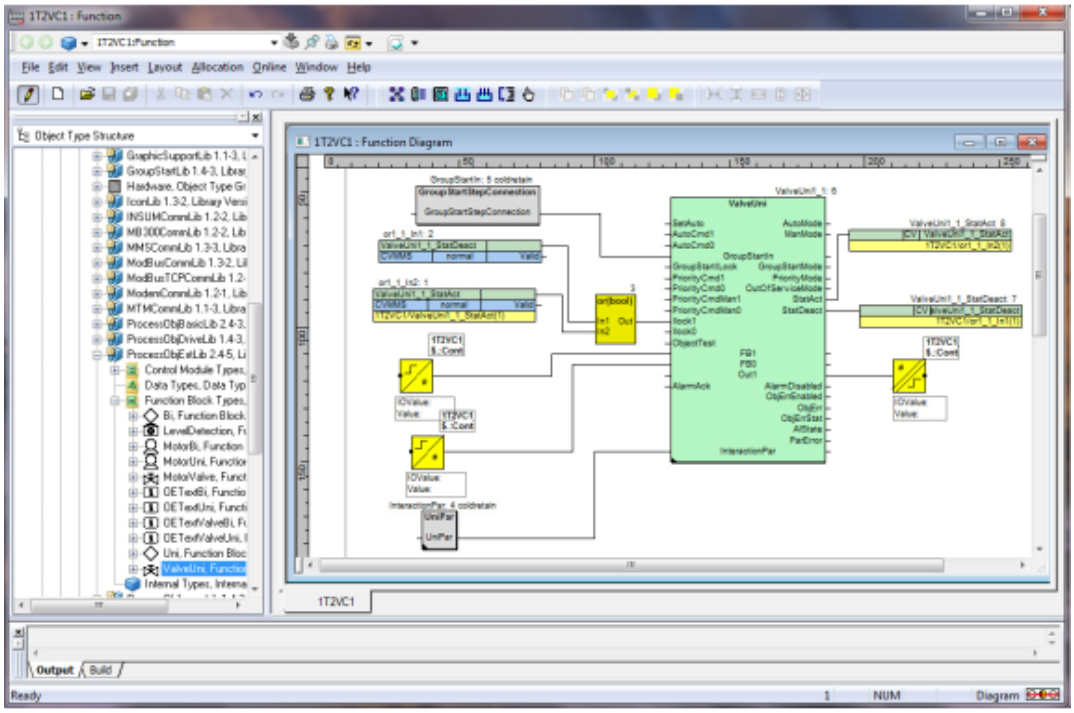
(Connected Libraries), hardware a tasks. Složka připojené aplikace (Connected Applications) obsahuje aplikace nahrané na kontroléru. Připojené knihovny (Connected Libraries) je složka s knihovnami používanými v rámci aplikace. Složka hardware zobrazuje ve stromové struktuře reálnou podobu hardwaru se všemi I/O kartami. Poslední složka Tasks, obsahuje tasky, které určují rychlost, a tudíž kolikrát bude diagram nebo program proveden (Obrázek 3).



Obrázek 3 Struktura složky Kontroléry

1.2 Grafické rozhraní Function Designer

Program Function Designer je součástí systému 800xA . Poskytuje grafické uživatelské rozhraní pro zjednodušení konstruování, dokumentace a údržbu komplexního řídicího systému AC800M.

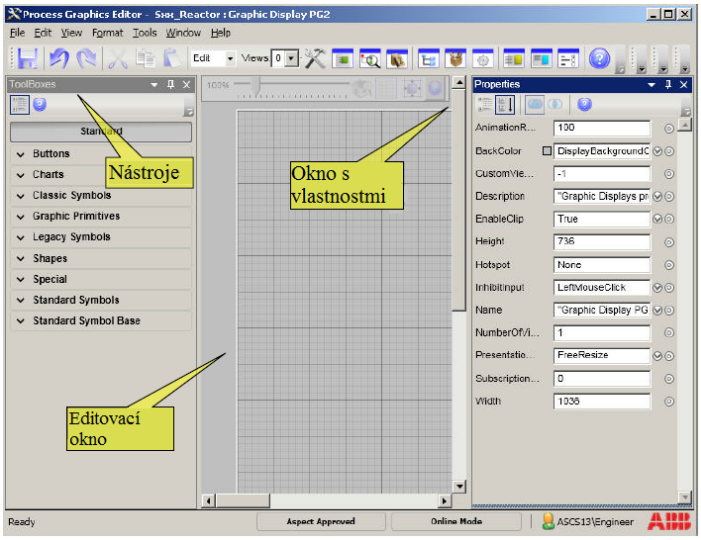


Obrázek 4 Prostředí programu Function Designer [ABB]

Function Designer používá funkční diagramy, které jsou propojené s funkčními aspekty objektu ve funkční struktuře (Obrázek 4). Může být zobrazen jako diagramy z funkčních aspektů. Uživatel může vložit a graficky spojit symboly bloků ve funkčním diagramu. Symbol bloku reprezentuje funkční komponent typu aspekt. Funkční Diagram může obsahovat kompletní řídicí smyčku procesu se všemi funkčními bloky, řídicími moduly, signály z Control Builderu a jejich propojeními. Také můžeme přidat textové poznámky, schématické obrázky atd. Do Funkčních diagramů můžeme vložit i sekvence se všemi kroky, podmínkami a akcemi.

1.3 Prostředí pro programování grafických objektů Graphic Builder

Graphic Builder je program potřebný pro vytváření grafických aspektů jednotlivých modulů. Mezi tyto aspekty patří například grafické displeje, grafické elementy používané na grafických displejích apod.

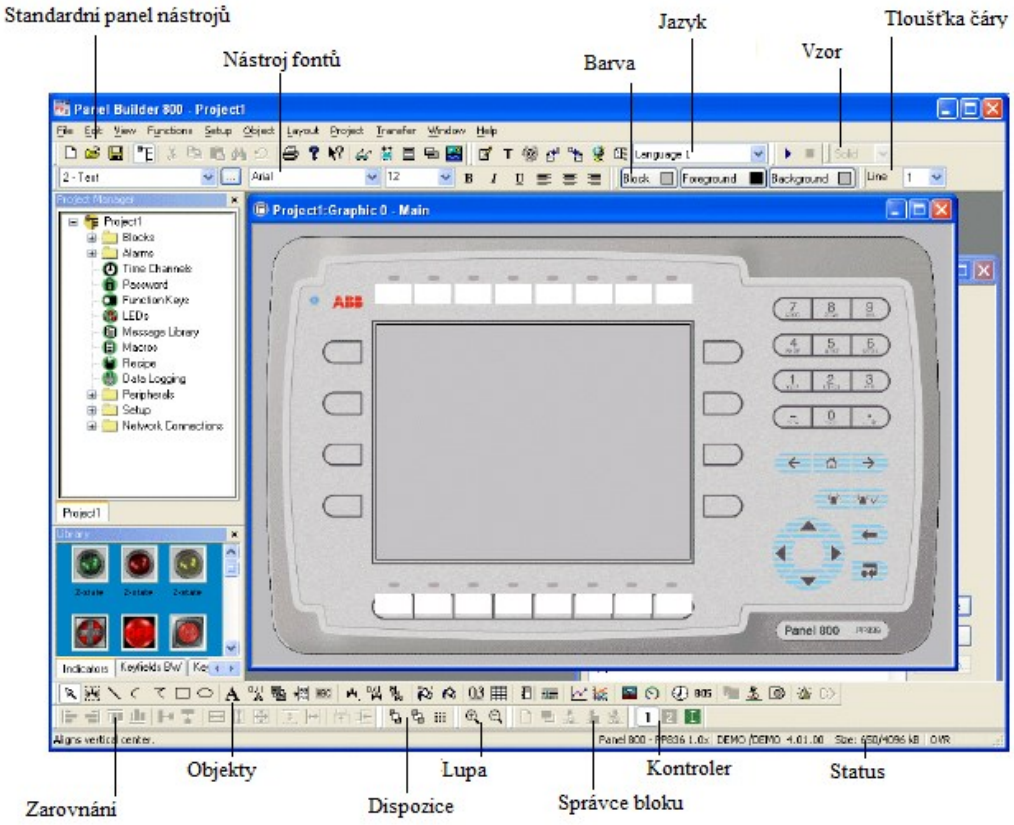


Obrázek 5 Hlavní okno Graphic Builderu [ABB - upraveno]

V okně Nástroje jsou základní obrazce, které můžeme použít, popřípadě se dá vyhledat námi vytvořený grafický element a přidat ho do editovacího okna. V editovacím okně vytváříme potřebný vzhled vizualizace. Při kliknutí na daný prvek se otevře okno s vlastnostmi prvku. Zde můžeme změnit například tvar, barvu a velikost prvku. Rovněž se dá danému prvku přiřadit funkce, kdy se má změnit určitá vlastnost prvku na základě hodnoty proměnné.

1.4 Program pro vytvoření vizualizace Panel Builder 800

Posledním programovacím nástrojem, který použiji, bude Panel Builder 800. V tomto prostředí se vytváří vizualizace operátorského panelu. Vytvoříme si grafiku, kterou chceme na panel umístit. Tato grafika bude zobrazovat proces, který řídíme. Můžeme u prvků nastavit změnu barvy podle stavu dané komponenty. Také můžeme přidat pole zobrazující hodnotu proměnné, například výšku hladiny v nádrži. Na panel můžeme přidat tlačítka k ovládání procesu. Výsledný panel bude sloužit ke komunikaci operátora s reálným hardwarem. Tématu tvoření vizualizace a komunikace operátora s hardwarem se budu podrobněji věnovat v kapitole 5.

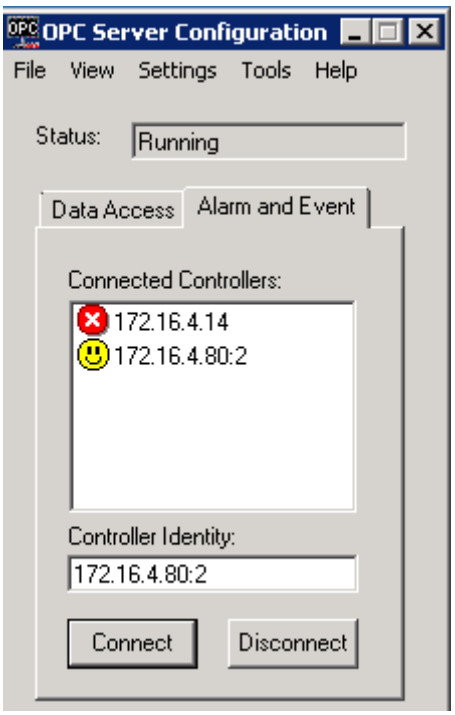


Obrázek 6 Program Panel Builder 800

1.5 Nastavení pro OPC Server 5.1.1

OPC je zkratka pro „OLE pro řízení procesů“, kde OLE znamená propojování a vkládání objektů. OPC poskytuje mechanismus pro poskytování dat ze zdroje dat a předání údajů do jakékoliv klientské aplikace. OPC má tři rozhraní: OPC přístup k datům v reálném čase, OPC alarm a událost pro data událostí a OPC historický přístup k datům pro historická data.

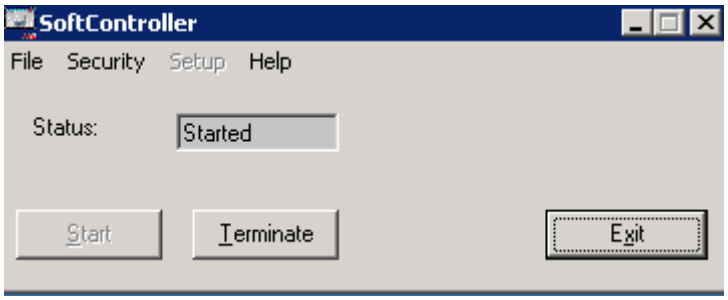
Komunikaci mezi OPC serverem a kontrolérem je nutno nastavit. V OPC serveru nastavíme do pole Control Identity IP adresu daného kontroléru a klikneme na tlačítko Připojit (Connect). Pokud se v pořádku připojí tak se nám zobrazí u IP adresy žlutý smajlík, v opačném případě červené kolečko s bílým křížkem. Toto připojení musíme udělat zvlášť pro přístup k datům a pro alarmy a události.



Obrázek 7 OPC Server

1.6 Simulovaný kontrolér SoftController 5.1.1

SoftController je program, jehož prostřednictvím můžeme vyzkoušet aplikaci vytvořenou v Control Builderu aniž bychom museli být napojeni na reálný hardware. SoftController spustíme stlačením tlačítka Start. Tím se změní stav kontroléru na Started. SoftController má stejnou IP adresu jako je IP adresa počítače. To využijeme, když chceme připojit OPC server. Nastavení je stejné jako u reálného kontroléru, pouze napíšeme IP adresu počítače a na konec adresy dáme dvojtečku a číslo 2. (např. 172.16.4.80:2)



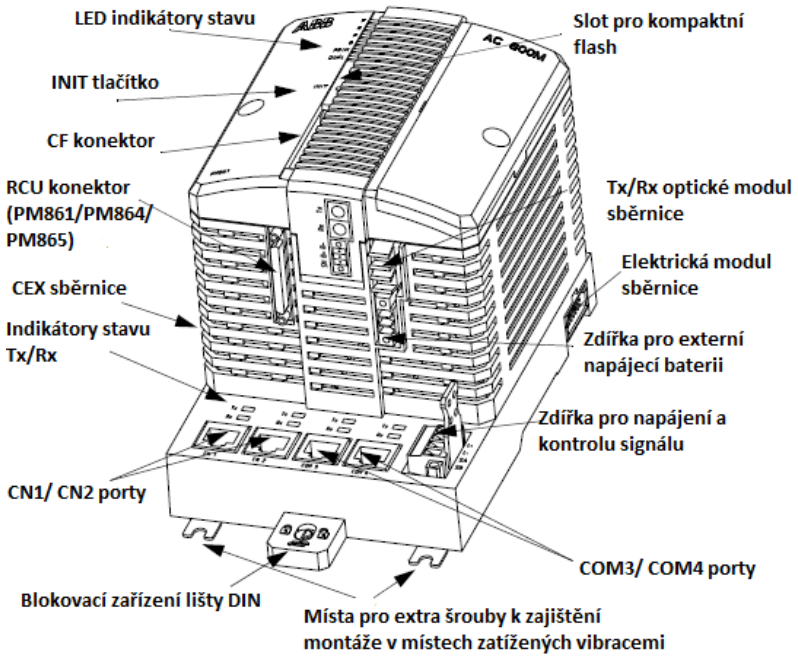
Obrázek 8 SoftController

2 Hardware a komunikace

Programovací nástroje používané v prostředí řídicího systému ABB 800xA jsou součástí hardwaru instalovaného v laboratoři. Na tomto hardwaru budu testovat komunikaci mezi HW simulátorem a kontrolérem a následně i komunikaci HW simulátoru s panelem ABB PP835.

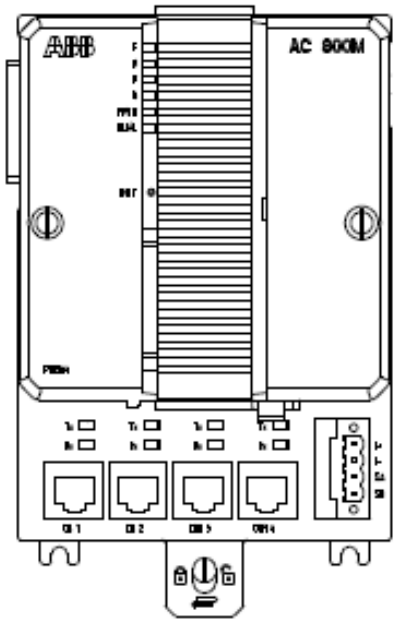
2.1 Řada kontrolérů AC800M

AC800M je hardwarová platforma skládající se z procesorové jednotky, komunikačního rozhraní a dalších zařízení. Po nahrání řídicího softwaru do kontroléru je možné AC800M používat jako samostatný regulátor nebo jako lokální regulátor v řídicím systému složeném z několika regulátorů, operačních stanic a serverů. Řídicí systém není s AC800M dodáván, je potřeba ho vytvořit v programu Control Builder M Professional. AC800M je složen z procesorové jednotky a komunikačních modulů, které jsou montovány na DIN lištu. K procesorové jednotce se mohou připojit I/O karty. Tyto karty se připojují přes ModuleBus.



Obrázek 9 Controller AC800M [ABB - upraveno]

K dispozici jsou různé procesorové jednotky, které se liší frekvencí a velikostí paměti. V našem případě jsme použili procesorovou jednotku PM864. Tato jednotka má paměť 32 MB RAM, frekvenci procesoru 96 MHz a je připevněna k základně TP830. Základna TP830 obsahuje porty pro připojení komunikačního rozhraní.



Obrázek 10 Procesorová jednotka PM864

Dále jsou k AC800M pomocí rozhraní CI854 připojeny I/O karty. V našem případě se jedná o 4 I/O karty, konkrétně o DI810, DO810, AI810 a AO820. Tyto karty slouží k propojování proměnných mezi hardwarem a softwarem v rámci projektu.

Specifikace I/O karet:

2.1.1 Digitální vstupní karta DI810

Digitální vstupní modul DI810 má 16 kanálů pro 24V DC digitální vstupy. Vstupy jsou rozděleny do dvou izolovaných skupin po 8 kanálech s kontrolou vstupního napětí pro každou skupinu.

Tabulka 1 Vlastnosti DI810

Vlastnost	**DI810**
Počet kanálů	16 (2x8)
Jmenovité napětí (rozsah procesorového napětí)	24V DC (18 až 30V DC)
Vstupní impedance	3,5 kΩ
Nominální vstupní napětí kanálu	6mA, 24 V DC
Ztrátový výkon	1,8 W

2.1.2 Digitální výstupní karta DO810

Digitální výstupní karta DO810 má 16 kanálů pro 24V DC digitální výstupy. Výstupy jsou rozděleny do dvou izolovaných skupin po 8 kanálech s napěťovou kontrolou pro jednotlivé skupiny. Výstupní rozsah napětí je od 10 do 30V a maximální trvalý výstupní proud je 0,5 A.

Tabulka 2 Vlastnosti DO810

Vlastnost	**DO810**
Počet kanálů	16 (2x8)
Rozsah napětí	12 – 32 V DC
Výstupní impedance	< 0.4Ω
Jmenovité izolační napětí	50V
Ztrátový výkon	2,1 W

2.1.3 Analogová vstupní karta AI810

Analogová vstupní karta AI810 má 8 proudových a napěťových vstupů. Vstupy jsou nezávislé na kanálu, ze kterého je napětí nebo proud měřeno.

Tabulka 3 Vlastnosti AI810

Vlastnost	**AI810**
Počet kanálů	8
Měřící rozsah	0…20mA, 0… 10V, 4…20mA, 2…10V
Vstupní napětí, maximální nedestruktivní	30V DC
Jmenovité izolační napětí	50V
Ztrátový výkon	1,5W

2.1.4 Analogová výstupní karta AO820

Analogová výstupní karta má 4 bipolární proudové a napěťové výstupy. Výběr mezi proudem a napětím je konfigurovatelný pro jednotlivé kanály.

Tabulka 4 Vlastnosti AO820

Vlastnost	**AO820**
Počet kanálů	4
Výstupní rozsah (nominální)	-20mA … +20mA, 0…20mA, 4…20mA nebo -10V…+10V, 0…10V, 2…10V
Výstupní zátěž, napěťové výstupy	≥ 5kΩ
Jmenovité izolační napětí	50V
Ztrátový výkon	6W

2.2 Hardwarový simulátor OSLO

Hardwarový simulátor OSLO je používán pro ověření správnosti a funkčnosti aplikace. Simulátor OSLO simuluje proces napouštění a vypouštění nádrže a přídavné funkce. Pro zobrazování stavu používá světelnou i zvukovou signalizaci. Například při otevření ventilu se nad příslušným ventilem rozsvítí zelené světlo a při zavření červené. Při zapnutí ventilátoru se na panelu ventilátor roztočí. Výšku hladiny v nádrži zobrazuje světelným napouštěním sloupce. Zvuková signalizace je určena především pro oznamování alarmů.

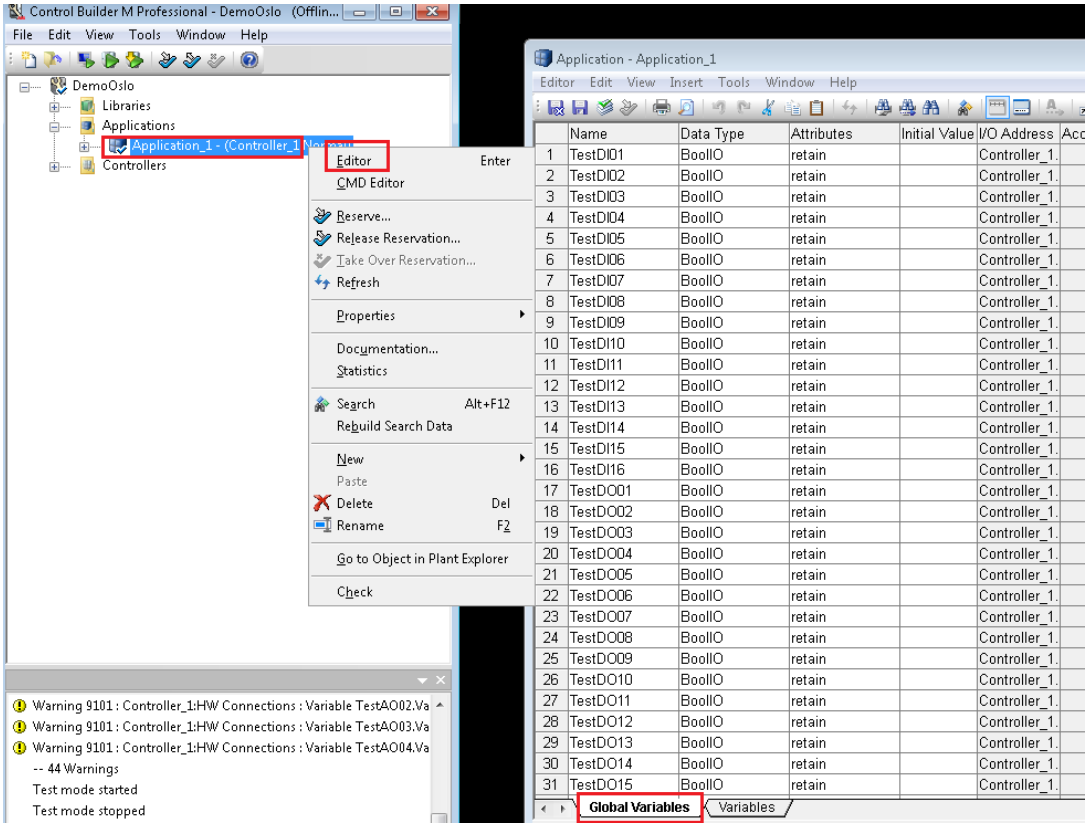
Jednotlivé prvky na modelu OSLO jsou propojeny dráty na sběrnici, od které jsou poté připojeny k jednotlivým kanálům vstupně-výstupních karet. Tyto karty jsou připojeny k procesoru kontroléru, do kterého je přes počítač nahrána aplikace.

Obrázek 11 Hardwarový simulátor OSLO

2.3 Otestování komunikace mezi HW simulátorem a kontrolérem AC800M

Pro otestování komunikace mezi HW simulátorem a kontrolérem AC800M jsme si v programu Control Builder M Professional vytvořili testovací projekt DemoOslo. Název DemoOslo vyjadřuje, že jde o testovací projekt a že pracujeme s hardwarovým simulátorem OSLO. Vytvořily se nám 3 základní složky: Knihovny (Libraries), Aplikace (Applications) a Kontroléry (Controllers).

Ve složce Aplikace (Applications) jsme vytvořili demo aplikaci, se kterou budeme pracovat. V editorovacím okně aplikace jsme nadefinovali globální proměnné. Vytváříme globální proměnné, protože s nimi potřebujeme pracovat v rámci celé aplikace a ne jen na jedné úrovni.

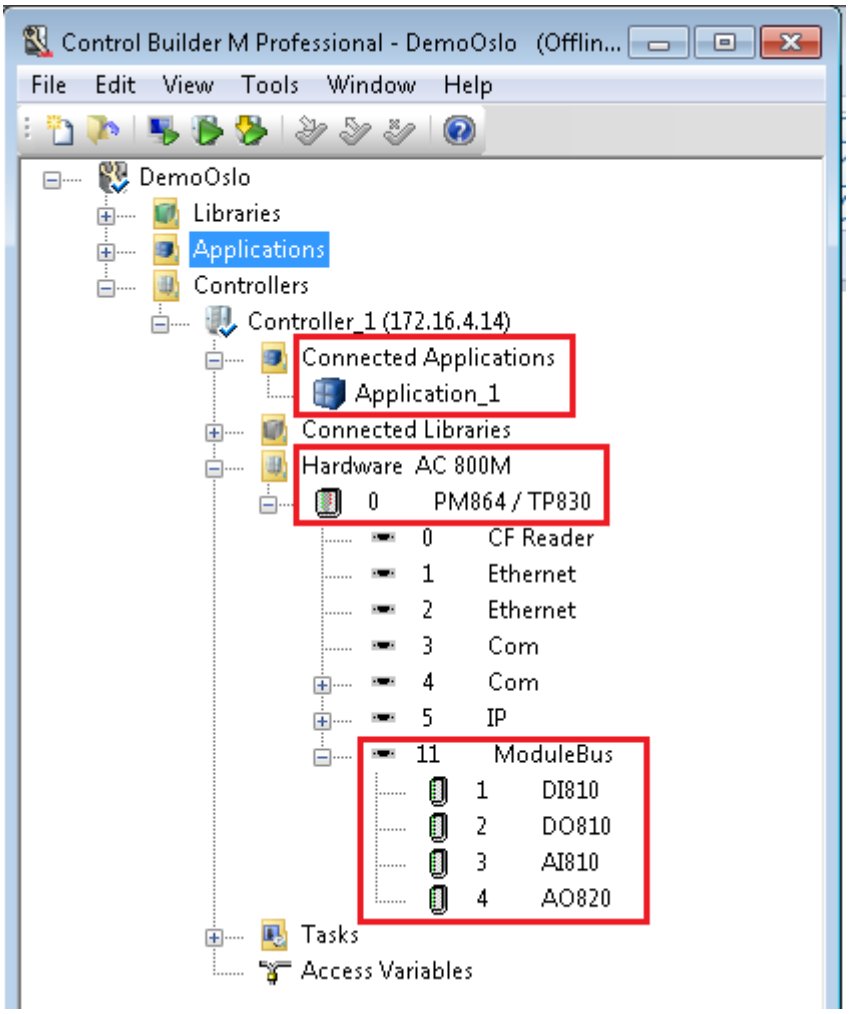


Obrázek 12 Demo aplikace a globální proměnné

Ve složce Kontroléry (Controllers) jsme si do podsložky Připojené aplikace (Connected Applications) připojili naši vytvořenou aplikaci. Nyní je potřeba nadefinovat strukturu reálného hardwaru do našeho projektu.

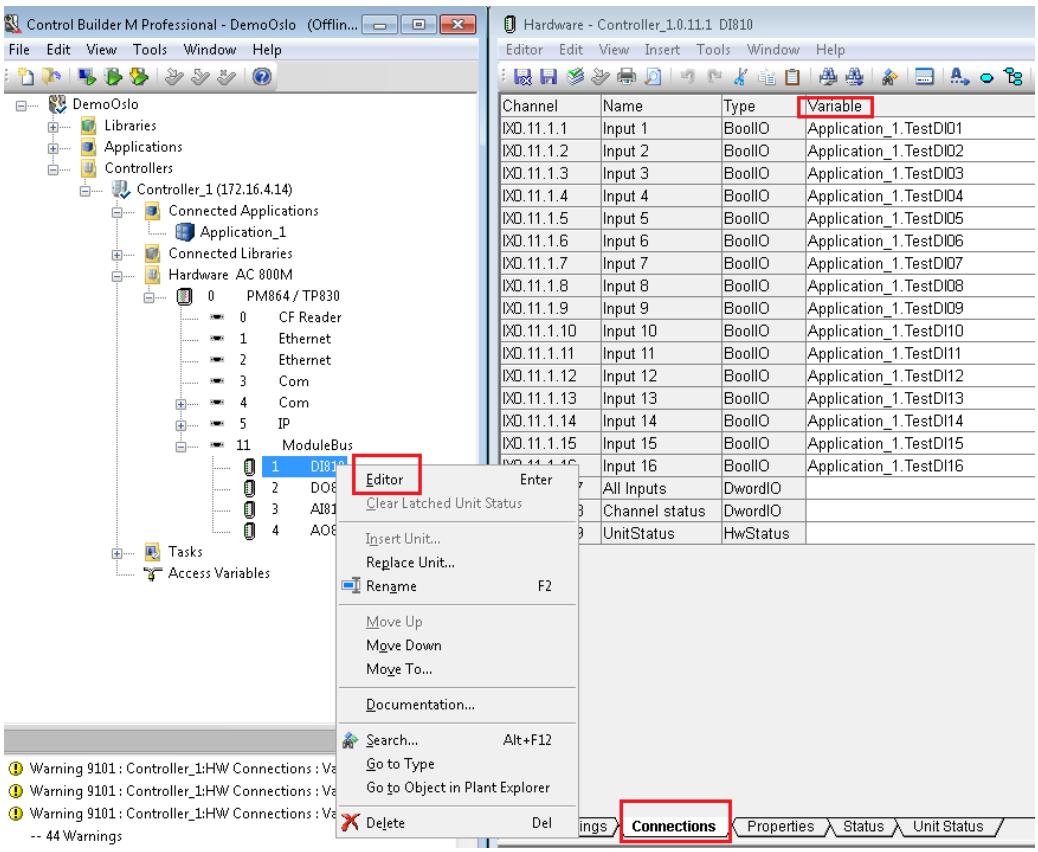
Vybereme složku Hardware AC800M a zde pomocí pravého kliknutí a příkazu Insert Units přidáme procesorovou jednotku PM864/TP830. Po rozevření vzniklé složky se nám zobrazí několik podsložek. My si vybereme podsložku s číslem 11, tedy

ModuleBus, protože tato sběrnice realizuje propojení I/O karet s kontrolérem. Pomocí pravého tlačítka a příkazu Insert Unit napojíme na sběrnici I/O karty v pořadí, v jakém jsou připojeny na reálném hardwaru.



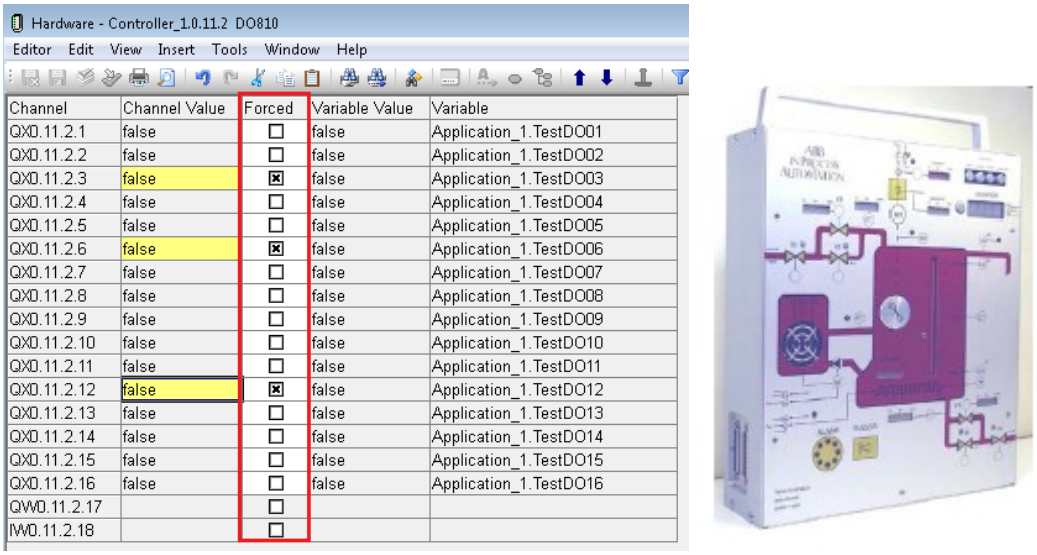
Obrázek 13 Připojení hardwaru

Po napojení všech I/O karet, je jednotlivě budeme otevírat a připojovat k nim příslušné globální proměnné z aplikace. Toto propojení se provádí v editorovacím okně, které otevřeme pravým tlačítkem myši. Okno má dole několik zarážek, my si vybereme Spojení (Connections) a zde přidáme do sloupce Proměnné (Variables) naše signály.



Obrázek 14 Připojení proměnných na I/O karty

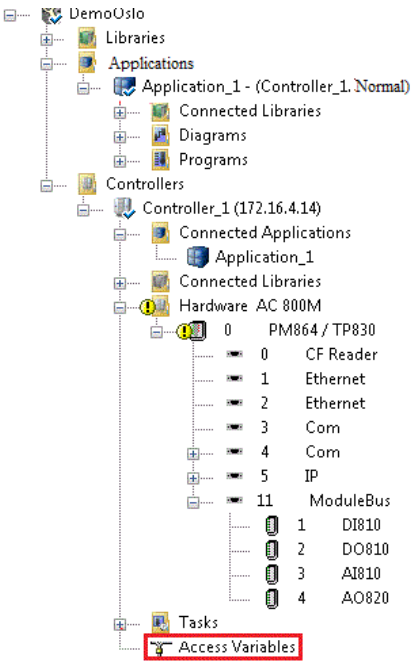
Abychom zjistili, na kterou pozici jednotlivých karet připojit dané proměnné, musíme si prostudovat tabulku signálů dodávanou k panelu Oslo. Každý signál má své číslo a pod tímto číslem ho lze vysledovat až k jedné z karet. Pro ověření, zda jsme vše napojili správně, nahrajeme aplikaci na kontrolér pomocí tlačítka Online v Control Builderu. Nyní můžeme v editorovacím okně každé karty přejít na záložku Status a zadat si ručně hodnoty pro jednotlivé signály. To znamená, že si ručně nastavíme, zda signál bude true nebo false. Poté již jen sledujeme, zda se na panelu Oslo projevila změna. Žlutá barva nám značí položku, kterou můžeme změnit.



Obrázek 15 Ruční zadávání hodnot a Panel OSLO

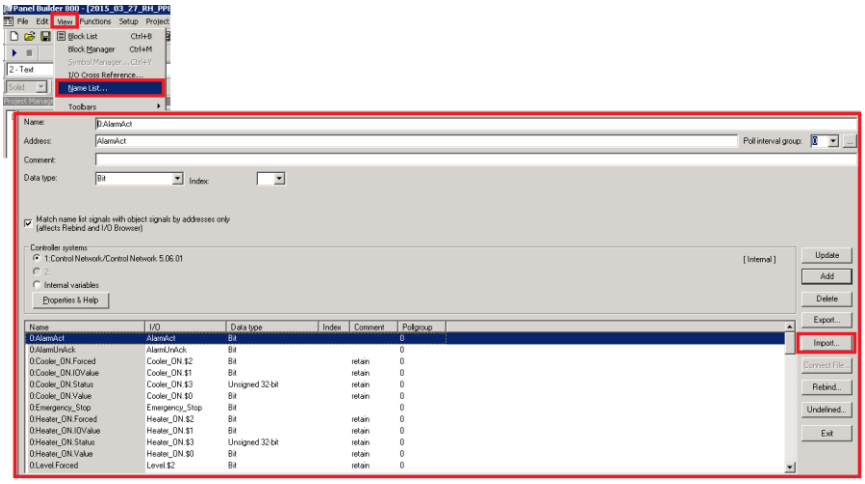
2.4 Komunikace mezi HW simulátorem a operátorským panelem PP835

Pro otestování komunikace HW simulátoru s operátorským panelem PP835 použijeme také aplikaci DemoOslo. Propojení proměnných se provádí v programu Control Builder, kdy ve složce Access Variables v záložce MMS vytvoříme proměnné, které se budou používat v rámci panelu PP835. K těmto proměnným připojíme ve sloupci Path námi vytvořené globální proměnné z aplikace.



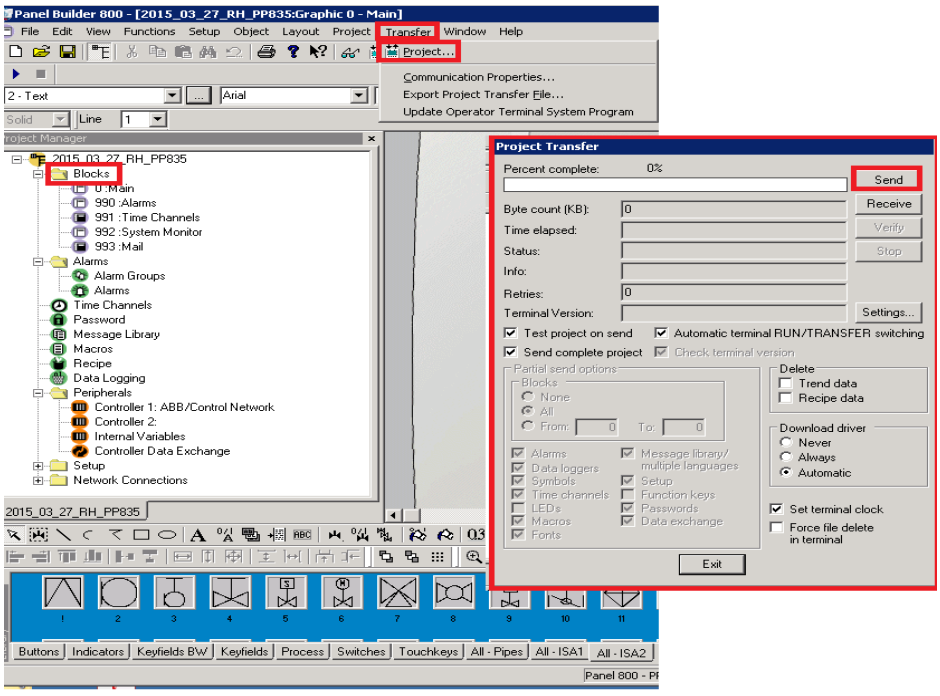
Obrázek 16 Access Variables

Nyní je potřeba tyto proměnné naimportovat do Panel Builderu 800. To provedeme následujícím způsobem, v horní liště vybereme záložku View -> Name list. A dáme Import.



Obrázek 17 Import Name Listu v Panel Builderu 800

K vytváření grafiky v Panel Builderu slouží záložka Blocks v levém menu a spodní lišta, ve které jsou předdefinované nejčastější prvky, například ventily, nádrže apod. Při vložení prvku se otevře tabulka s vlastnostmi prvku. Zde připojíme proměnnou, kterou jsme si vytvořili v Access Variables a vybereme vzhled prvku pro stav True a pro stav False. Pro otestování funkčnosti komunikace klikneme na záložku Transfer v horní liště, vybereme Project a pošleme data do reálného kontroléru pomocí tlačítka Send. Nyní se nám v grafice budou zobrazovat stavy jednotlivých komponent.



Obrázek 18 Nahrávání projektu do kontroleru

3 Funkční popisy stávajících knihoven

3.1 Funkční popis knihovny

Funkční popis knihovny je papírová dokumentace, která je vytvářena k jednotlivým knihovnám. V této dokumentaci vidíme strukturu celé knihovny, připojené jiné knihovny, ze kterých bylo čerpáno, dále datové typy, pokud byly použity a především kontrolní moduly, které jsou důležité pro vytváření následné aplikace.

V rámci datových typů vidíme tabulku všech proměnných, které obsahují. U každé proměnné je nadefinováno její jméno, datový typ a popis, k čemu je tato proměnná používána.

	Name	Data Type	Attributes	Initial value	ISP value	Description
1	OverloadWarningStatus	bool	retain	false		Warning Overload
2	RunningHours	time	retain	0s		Running Hours
3	Current	real	retain	0		Current
4	CurrentUnit	string[4]	coldretain	'A'		Current Units
5	CurrMax	dint	retain			Max range of the current value
6	CurrMin	dint	retain			Mix range of the current value
7	AckCondState_GF	dint	retain			OUT NONSIL ALARM AlCondState (0:AlNotDefined, 1:Disabled, 2:Idle, 3:InactiveUnacked, 4:ActiveAcked, 5:ActiveUnacked, 6:AutoDisabled)
8	AckCondState_W	dint	retain			OUT NONSIL ALARM AlCondState (0:AlNotDefined, 1:Disabled, 2:Idle, 3:InactiveUnacked, 4:ActiveAcked, 5:ActiveUnacked, 6:AutoDisabled)

Obrázek 19 Ukázka části datového typu knihovny Sevan300VMSLib [ABB]

U jednotlivých kontrolních modulů je zdokumentováno editorovací okno se všemi proměnnými použitými v rámci daného modulu. U každé proměnné je nadefinováno jméno, datový typ, směr (in/out) a popis. Dále jsou zobrazeny tabulky s parametry a funkčními bloky používanými v kontrolním modulu.

	Name	Data Type	Attributes	Initial value	Description
1	vCondnameOverlAlm	string[20]	retain	'GW'	Condition for Overload alarm name
2	vMessageOverlAlm	string[20]	retain	'General Warning'	Message when Overload alarm active
3	vCondnameGenAlm	string[20]	retain	'OW'	Condition for General alarm name
4	vMessageGenAlm	string[20]	retain	'Overload Warning'	Message when General alarm active
5	vEnableEv	bool	retain		Enable Events
6	vEnableAL	bool	retain	0	Enable Alarm
7	vLSB	dword	retain	0	variable for current calculation
8	vMSB	dword	retain	0	variable for current calculation
9	vAvailable	bool	retain	true	Available

Obrázek 20 Ukázka vnitřních proměnných kontrolního modulu v knihovně Sevan300VMSLib [ABB]

V kontrolních modulech se vytvářejí funkce, které má modul vykonávat. Například se naprogramuje, které proměnné jsou vstupní a které výstupní, kdy se mají zobrazovat alarmy a události atd. Tyto kódy jsou rovněž zobrazeny ve funkčních popisech.

```
(*Enable Disable Alarm*) (**)

if PDA = 2 then
  vEnableAL:=1;
  vEnableEv:=1;
else if PDA = 1 then
  vEnableAL:=0;
  vEnableEv:=1;
else
  vEnableAL:=0;
  vEnableEv:=0;
```

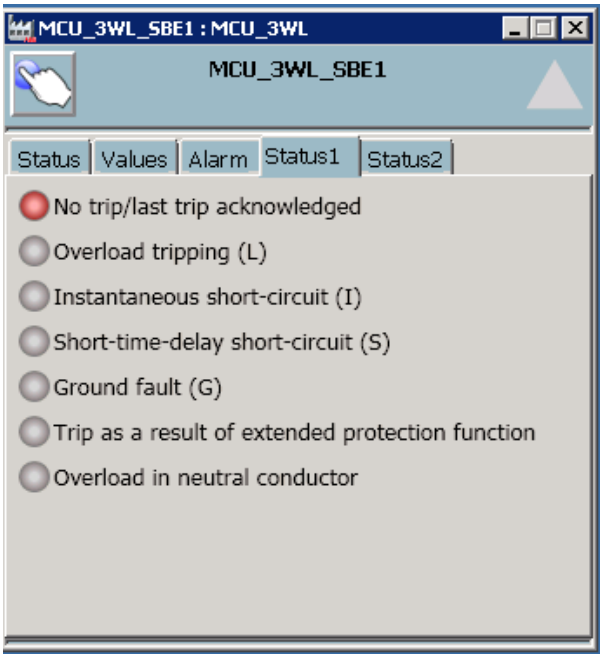
Obrázek 21 Ukázka kódu pro vygenerování alarmu a události pro kontrolní modul v knihovně Sevan300VMSLib [ABB]

Pro jednotlivé kontrolní moduly se vytvářejí také grafické elementy, které znázorňují funkci daného modulu (polohu, stav, atd.). Pro tyto grafické elementy se v knihovnách vytvářejí parametry označované buď HMI nebo HSI. Těmto parametrům se následně přiřazují vnitřní proměnné modulu.

```
(*Connection between HSI and local variables*)

HSI.Current := vCurrent;
HSI.OverloadWarningStatus := vOverloadWarningStatus;
HSI.General_Fault := vGeneralFault;
HSI.Available := vAvailable;
HSI.RunningStatus := vRunningFeedback;
HSI.Remote_Local := Z.LO;
HSI.RunningHours:=vRunningHours;
HSI.CurrentUnit:=CurrUnit;
```

Obrázek 22 Ukázka propojení HSI proměnných s vnitřními proměnnými v knihovně Sevan300VMSLib [ABB]

Funkční popisy knihoven rovněž zobrazují grafiku vytvořenou k jednotlivým modulům. Na obrázku 23 je vidět ukázka komplexnějšího Faceplate s několika záložkami jako je Status, Values, Alarm a záložky se zvýrazněnými aktuálními stavy.

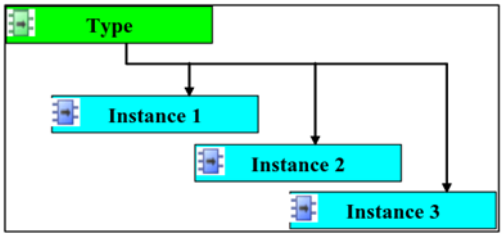


Obrázek 23 Ukázka Faceplate MCU_3WL z knihovny Sevan300VMSLib [ABB]

Knihovny se ve firmě ABB uchovávají několik let, protože v rámci různých projektů se některé prvky opakují. Každou knihovnu lze upravit a rozšířit. Každá oblast (papíry, minerály,…) v rámci firmy ABB má své knihovny, které obnovuje, vylepšuje a opakovaně používá. Tento systém urychluje práci při vytváření projektu.

3.2 Knihovna

Při vytváření aplikace se může stát, že potřebujeme použít nějaký modul několikrát, například ventil. Pro takové případy se používají knihovny. V knihovnách si vytváříme typy, které se mohou používat dále v celém projektu jako instance. Výhoda vytváření typů a instancí spočívá v tom, že když se objeví v nějaké instanci chyba, tak ji stačí opravit v odpovídajícím typu. Veškeré změny uskutečněné v typu se automaticky přenesou na všechny instance tohoto typu. V Control Builderu se rozlišují typy a instance barevně. Typy jsou zelené a instance modré (Obrázek 24).



Obrázek 24 Barevné odlišení typu a instance

Vytvořené knihovny jsou uspořádány do stromové struktury, která obsahuje několik složek:

- Připojené knihovny (Connected Libraries)

V této složce můžeme k naši knihovně připojit již vytvořené knihovny a používat jejich typy.

- Datové typy (Data Types)

Datové typy se používají při deklarování proměnné uvnitř aplikace. Firma ABB má definovanou sadu standardních datových typů ve svých standardních knihovnách. Složka Datové typy slouží především k vytváření vlastních datových typů. Datové typy vytvořené v knihovně jsou přístupné všem aplikacím a knihovnám, které mají tuto knihovnu připojenu.

Mezi základní, předdefinované datové typy patří bool, int, string, real apod. Mezi strukturované datové typy patří BoolIO a RealIO.

- Typy kontrolních modulů (Control Module Types)

V této složce programujeme jednotlivé typy kontrolních modulů. Obsahují kód, grafiku, jiné funkční bloky atd.

- Typy funkčních bloků (Function Block Types)

Typy funkčních bloků se používají pro vytváření dalších funkcionalit pro programátora. Ve standardních knihovnách firmy ABB najdeme množství předdefinovaných typů funkčních bloků, například pro alarmy, ventily, motory apod.

- Diagramové typy (Diagram Types)

Mohou obsahovat další diagramy, které jsou instancemi diagramových typů, například funkce, instance funkčních bloků a instance kontrolních modulů. V editoru diagramu se mohou přidávat objekty, které se dají poté graficky zpracovat pro vytvoření logiky.

Status knihovny

Knihovna může mít 3 statusy, které popisují, jak můžeme knihovnu použít.

- Open

Tento stav značí, že obsah knihovny může být změněn. Tento stav mají knihovny, které jsou ve vývoji.

- Closed

V tomto stavu nemůžeme v knihovně nic měnit, ale stále se dá její stav změnit zpět do stavu Open, abychom mohli nějakou změnu provést.

- Released

Nemůžeme měnit obsah knihovny. Jestliže je nutné nějakou změnu provést, musí se vytvořit nová verze této knihovny.

Pro přehlednost bych shrnula statusy knihovny do tří vět: Pokud je knihovna ve stavu Open, může se změnit do stavu Closed nebo Released. Pokud je knihovna ve stavu Released, může se změnit do stavu Open jedině vytvořením nové verze této knihovny. A je-li knihovna ve stavu Closed, může se změnit na Open nebo Released.

4 Návrh a vytvoření knihovny

Jak již bylo řečeno, knihovny se používají pro vytváření základních typů kontrolních modulů, které bude potřeba v rámci aplikace použít. V knihovně se vytvoří základní typ a pak jej v aplikace mohu několikrát použít jako instanci. Například je nutné v rámci projektu mít 6 ventilů. Každý z nich bude mít rozdílnou funkci, ale jedná se o stejný typ i se stejným grafickým zobrazením. Abych nemusela vytvářet ventil šestkrát, tak si v knihovně vytvořím jeden typ s kódy i grafikou a v aplikaci použiji šest instancí tohoto typu.

4.1 Návrh knihovny

V rámci mého projektu budu potřebovat vytvořit 6 základních typů a to: Digitální vstupní signál (DIS), digitální výstupní signál (DOS), analogový vstupní signál (AIS), analogový výstupní signál (AOS), motor a ventil (Valve). Těchto 6 typů je vybráno, protože v projektu vytvořím simulátor nádrže, ve kterém budu potřebovat ventily pro napouštění a vypouštění tekutiny, modul AIS bude použit pro zobrazování hladiny a teploty v nádrži, DOS bude použit pro ohřívání tekutiny a motor pro chlazení a míchadlo. Pro grafickou simulaci hardwaru bude potřeba vytvořit pro jednotlivé prvky tzv. Faceplaty. Faceplate je v podstatě obrazovka, na které jsou jednotlivá tlačítka, kterými můžeme daný prvek ovládat (ON/OFF, Manuální/Automatické ovládání apod.). Na Faceplate se rovněž vytváří grafický element, který indikuje stav daného prvku pomocí předurčených barev.

Nejprve si určím vlastnosti jednotlivých prvků, které jsem popsala výše.

- DIS = digitální vstupní signál

Modul DIS musí umět číst signál z hardwaru, zobrazovat hodnotu tohoto signálu, detekovat a zobrazovat alarmy a eventy podle systému 0/1/2, tzn., že při 0 není zobrazen ani alarm ani event, při 1 se zobrazuje pouze event a při 2 hlásí alarmy i eventy. Dále máme za úkol k tomuto CM vytvořit Faceplate, na kterém se zobrazí stav hardwaru, alarmy a bude zde záložka pro ruční zadávání hodnot.

- DOS = digitální výstupní signál

Kontrolní modul DOS bude umožňovat zadávání výstupní hodnoty signálu.

- AIS = analogový vstupní signál

Kontrolní modul AIS bude podobný kontrolnímu modulu DIS. Bude umět číst signál z hardwaru, zobrazovat jeho hodnotu, detekovat a zobrazovat alarmy a eventy. Přičemž kontrolní modul AIS má 4 úrovně alarmů: HH = high high, H = high, L = low, LL = low low. Pro každou úroveň alarmů bude detekovat alarmy a eventy dle 0/1/2. Kontrolní modul bude mít Faceplate, na kterém bude zobrazena hodnota hardwarového signálu, alarmy a úrovně jednotlivých alarmů. Dále bude umožněno ruční zadávání hodnot přímo z Faceplatu a zadávání úrovně alarmů z Faceplatu.

- AOS = analogový výstupní signál

Kontrolní modul AOS bude stejný jako DOS, rovněž bude umožňovat zadávat výstupní hodnoty signálu.

- Valve = ventil

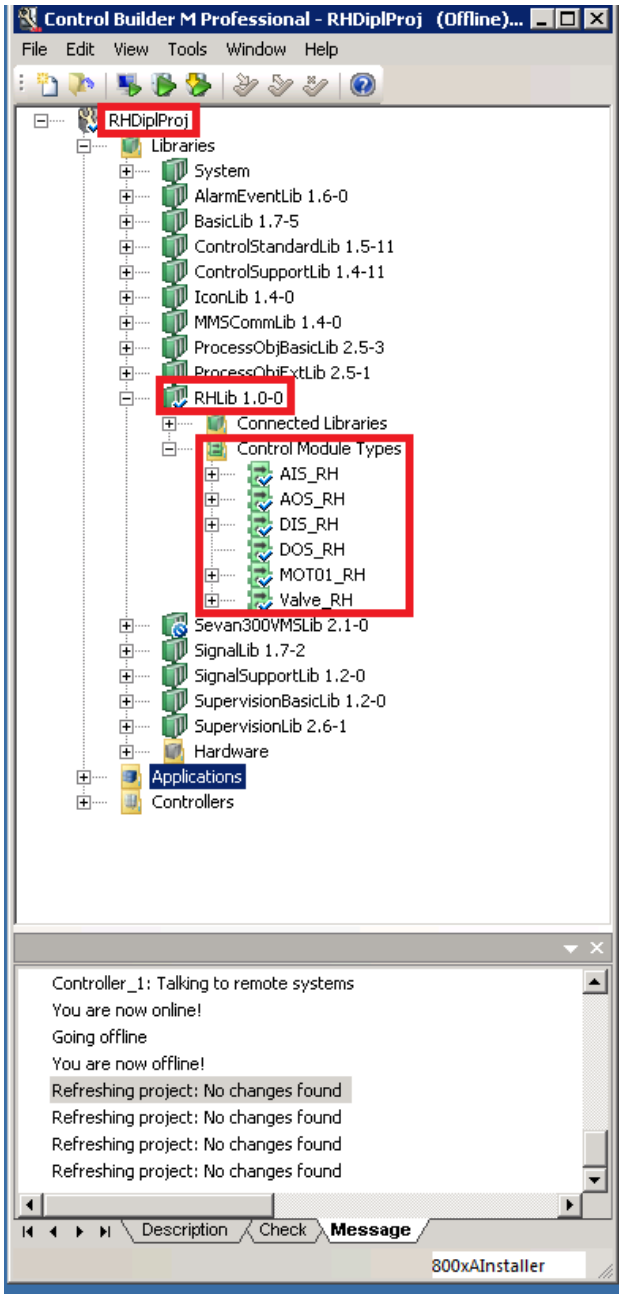
Dalším kontrolní modulem bude ventil. Ventil bude typu Open/Close. Bude detekovat alarm, pokud nebude mít feedback. Na Faceplatu bude zobrazen status ventilu a alarm.

- Motor

Kontrolní modul Motor bude typu Start/Stop a na Faceplatu bude zobrazovat status motoru a jeho signálů. A stejně jako Ventil bude detekovat a zobrazovat alarm, pokud nebude mít feedback.

4.2 Vytvoření knihovny

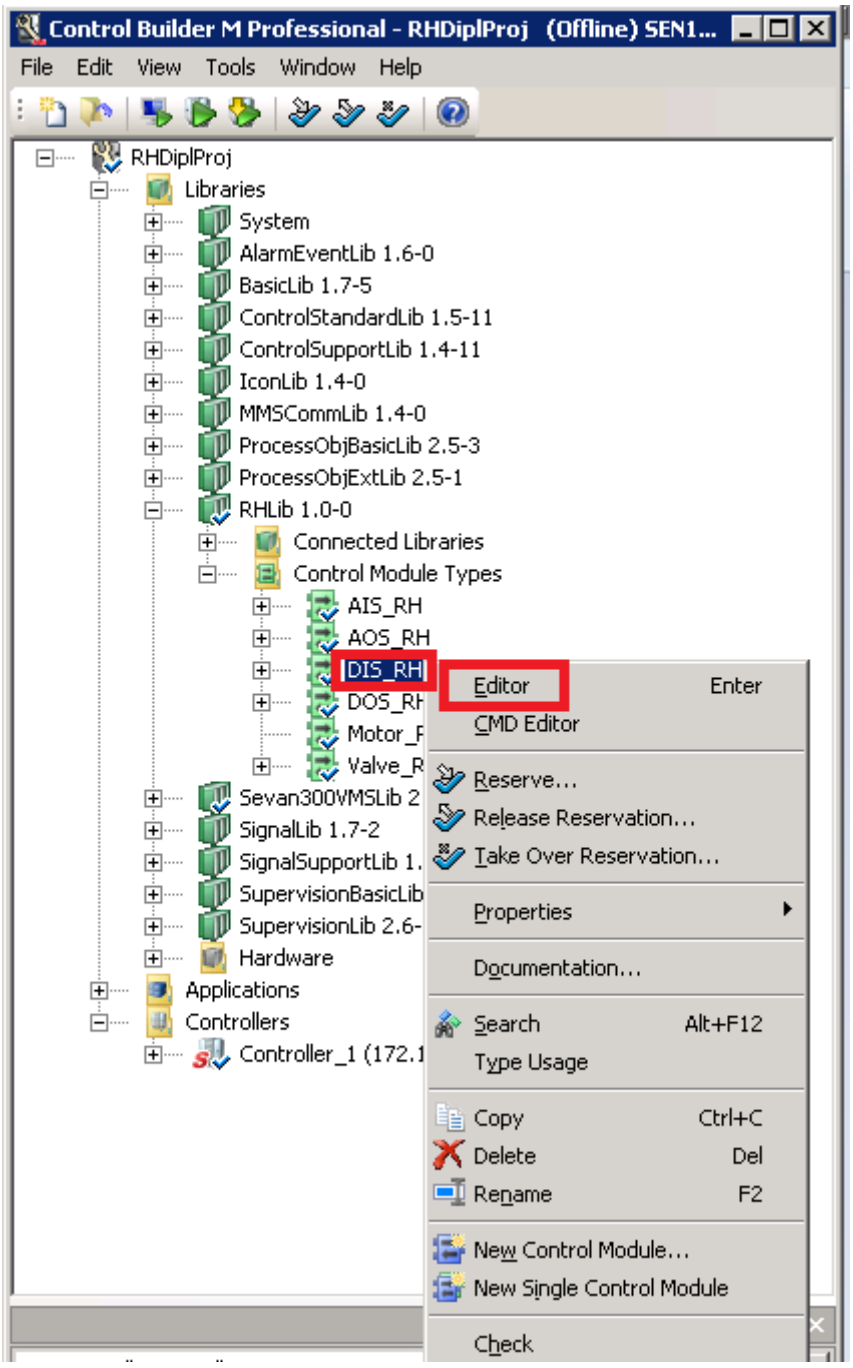
Nyní si vytvořím knihovnu s výše popsanými moduly. Začnu tím, že v programu Control Builder vytvořím nový projekt s názvem RHDiplProj. V tomto projektu si ve složce Knihovny (Libraries) vytvořím novou knihovnu s názvem RHLib. V rámci knihovny si ve složce Kontrolní moduly (Control Module Types) založím jednotlivé moduly (Obrázek 25).



Obrázek 25 Vytvořená knihovna s moduly

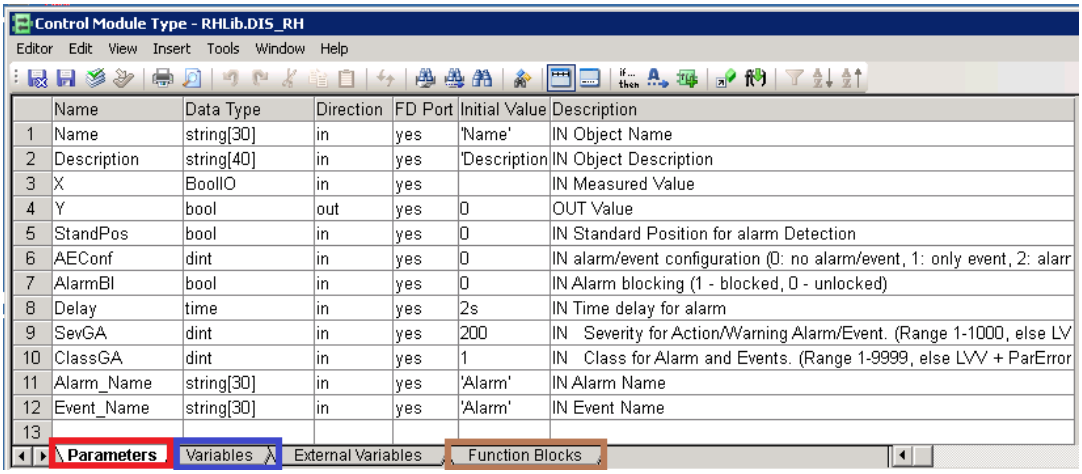
4.2.1 Digitální vstupní signál (DIS)

Začnu vytvořením kontrolního modulu DIS. Budu muset nadeklarovat proměnné, parametry a napsat kódy pro funkcionalitu modulu. Nejprve vybereme kontrolní modul DIS a pravým tlačítkem otevřeme Editor.



Obrázek 26Otevření Editoru pro kontrolní modul DIS

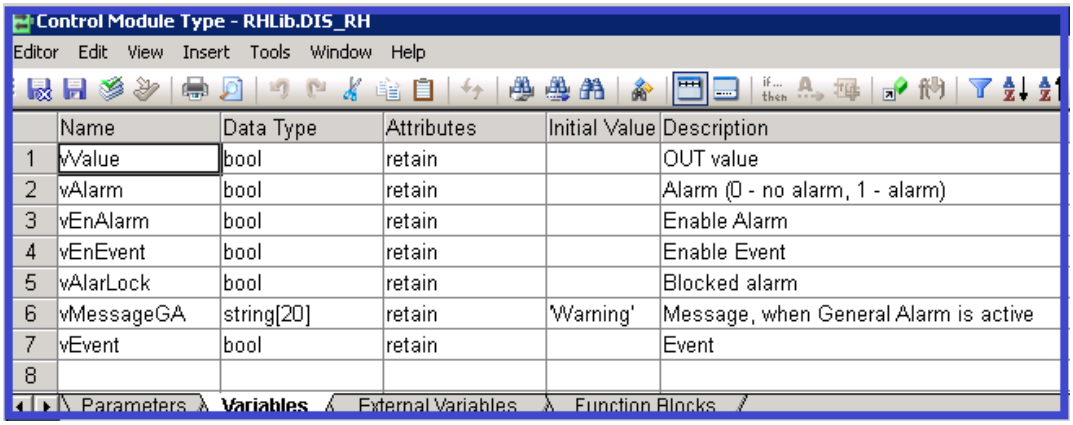
Otevře se nám okno, ve kterém budeme deklarovat potřebné proměnné a psát kód.



	Name	Data Type	Direction	FD Port	Initial Value	Description
1	Name	string[30]	in	yes	'Name'	IN Object Name
2	Description	string[40]	in	yes	'Description'	IN Object Description
3	X	BoolIO	in	yes		IN Measured Value
4	Y	bool	out	yes	0	OUT Value
5	StandPos	bool	in	yes	0	IN Standard Position for alarm Detection
6	AEConf	dint	in	yes	0	IN alarm/event configuration (0: no alarm/event, 1: only event, 2: alarr
7	AlarmBl	bool	in	yes	0	IN Alarm blocking (1 - blocked, 0 - unlocked)
8	Delay	time	in	yes	2s	IN Time delay for alarm
9	SevGA	dint	in	yes	200	IN Severity for Action/Warning Alarm/Event. (Range 1-1000, else LV
10	ClassGA	dint	in	yes	1	IN Class for Alarm and Events. (Range 1-9999, else LVV + ParError
11	Alarm_Name	string[30]	in	yes	'Alarm'	IN Alarm Name
12	Event_Name	string[30]	in	yes	'Alarm'	IN Event Name
13						

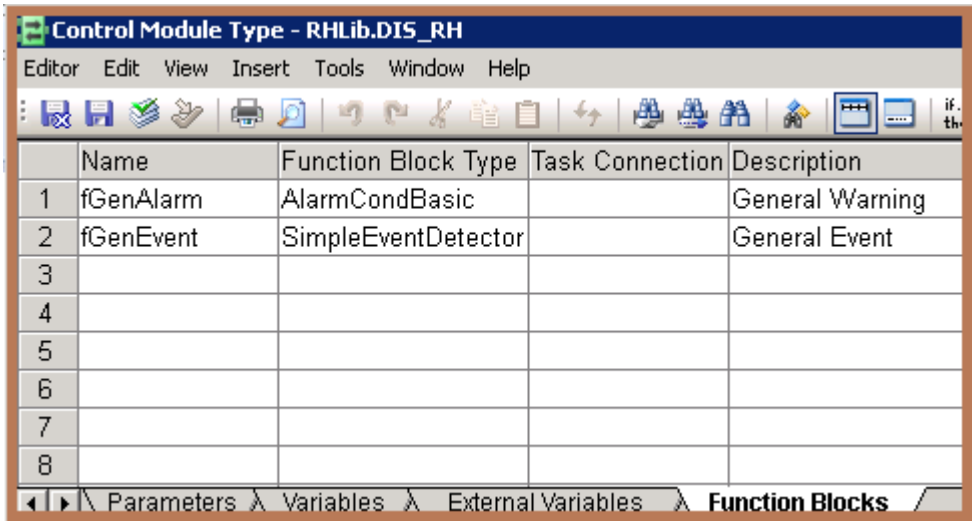
Obrázek 27 Okno Editor pro modul DIS

Na obrázku 27 můžeme vidět nadeklarované parametry a další záložky jako jsou Proměnné (Variables), Externí proměnné (External variables) a Funkční bloky (Function Blocks). V rámci knihoven se externí proměnné nepoužívají, proto nejsou označeny barevně. V záložce Parametry (Parameters) jsou hlavní vlastnosti modulu, jako název, popis, vstupní veličina, výstupní veličina atd. V tabulce vidíme několik sloupců, tyto sloupce je potřeba vyplnit: do sloupce Datový typ (Data Type) vypisujeme datový typ proměnné, do sloupce Směr (Direction) napíšeme, zda se jedná o vstupní nebo výstupní proměnnou (in/out), do sloupce Výchozí hodnota (Initial Value) můžeme, ale nemusíme napsat výchozí hodnotu proměnné a sloupec Popis (Description) obsahuje popis jednotlivých parametrů pro upřesnění. Jednotlivé záložky lze vidět na obrázcích 28 a 29.



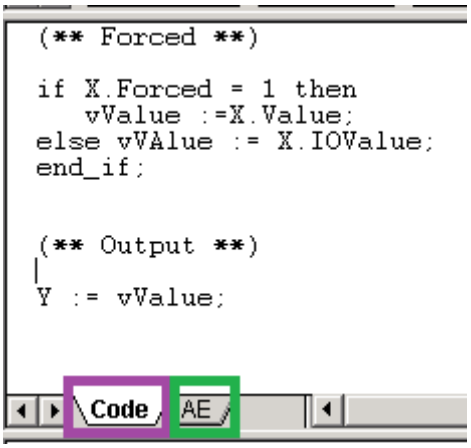
	Name	Data Type	Attributes	Initial Value	Description
1	vValue	bool	retain		OUT value
2	vAlarm	bool	retain		Alarm (0 - no alarm, 1 - alarm)
3	vEnAlarm	bool	retain		Enable Alarm
4	vEnEvent	bool	retain		Enable Event
5	vAlarLock	bool	retain		Blocked alarm
6	vMessageGA	string[20]	retain	'Warning'	Message, when General Alarm is active
7	vEvent	bool	retain		Event
8					

Obrázek 28 Záložka Variables u modulu DIS



Obrázek 29 Záložka Function Blocks u modulu DIS

V okně Editoru se nachází záložka pro psaní kódů. Na obrázku 30 můžeme vidět jednoduchý kód pro přiřazení proměnné parametru. A mírně složitější kód pro ruční zadávání výstupní proměnné.



Obrázek 30 Kód pro výstupní proměnnou a ruční zadávání hodnot modulu DIS

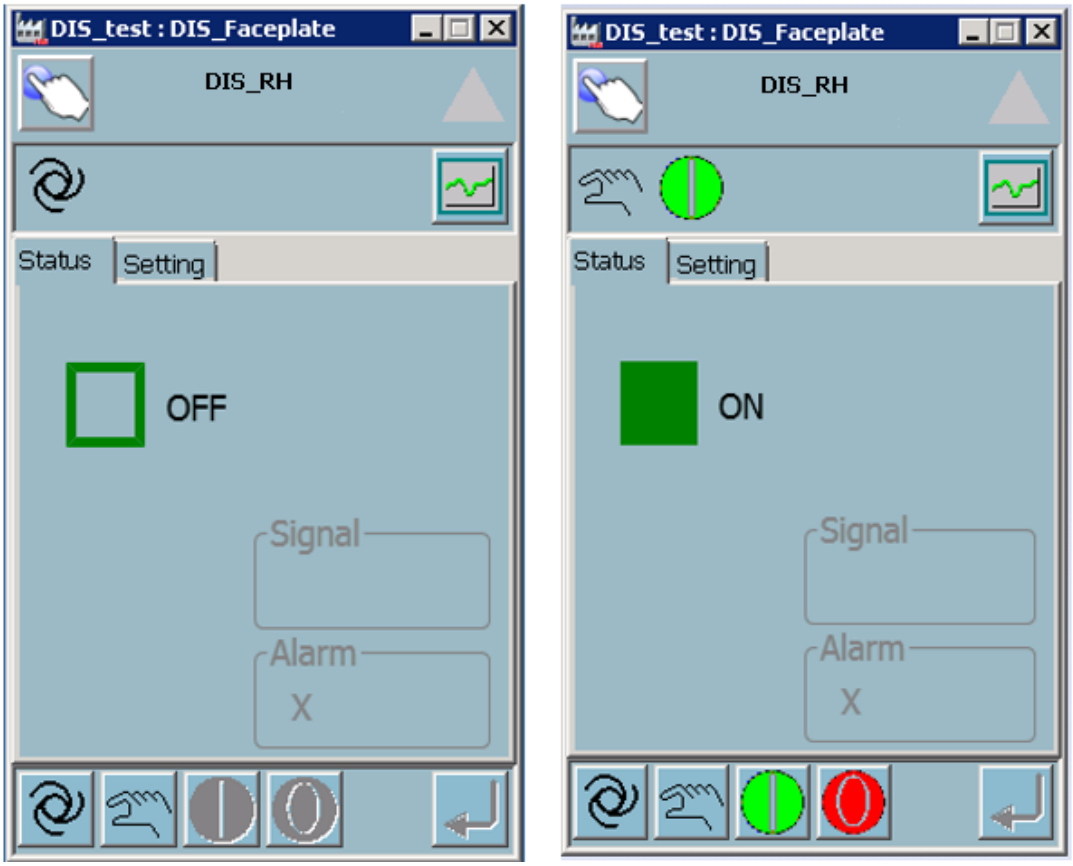
Na obrázku 31 je otevřena záložka AE (Alarm Event). Na této stránce je napsán kód pro konfiguraci a spuštění alarmů a událostí (eventů).

```
If vValue then
      if StandPos = false then
         vAlarm := true;
      else
         vAlarm := false;
      end_if;
else
   if StandPos= false then
      vAlarm := false;
   else
      vAlarm := true;
   end_if;
end_if;


(** AE Configuration **)


if AEConf = 2 then
   vEnAlarm := 1;
   vEnEvent := 1;
else if AEConf = 1 then
   vEnAlarm :=0;
   vEnEvent :=1;
else
   vEnAlarm := 0;
   vEnEvent := 0;
end_if;
end_if;


(** Blocking alarm **)

if AlarmBl then
   vAlarLock :=1;
else
   vAlarLock :=0;
end_if;

(** Function blocks - AE **)

fGenAlarm( Signal := vAlarm,
           SrcName := Alarm_Name,
           Message := vMessageGA,
           Severity := SevGA,
           Class := ClassGA,
           EnDetection := vEnAlarm & not vAlarLock );

fGenEvent ( Signal := vEvent,
            SrcName := Event_Name ,
            Message := vMessageGA,
            Severity := SevGA,
            Class := ClassGA,
            EnDetection := vEnEvent );
```

Code AE

Obrázek 31 Kód pro alarmy a události modulu DIS

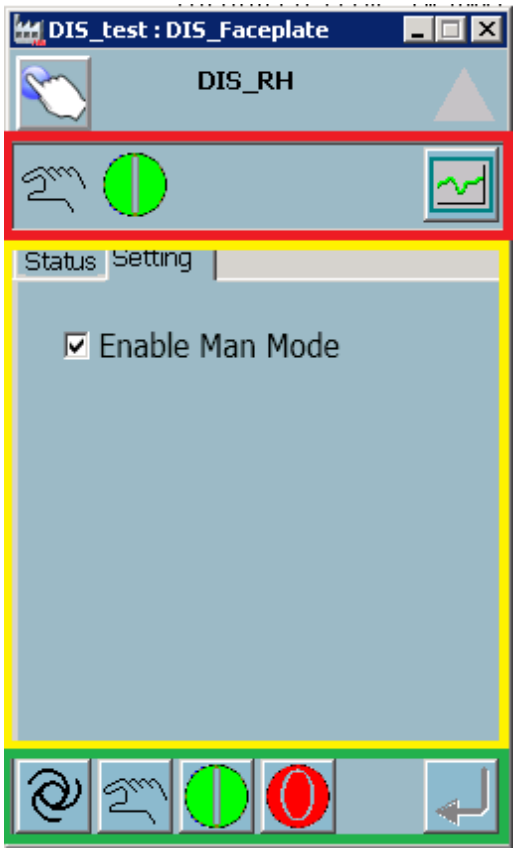
Pro každý vytvořený kontrolní modul je třeba vytvořit faceplate. Faceplate se vytváří v prostředí Plant Explorer Workplace. Pro modul DIS budeme potřebovat signalizaci pro stav hardwarového signálu, ruční zadávání hodnot a alarmy. Jednotlivé prvky je třeba vytvořit jako grafické a faceplate elementy, následně jim v grafickém editoru dát konkrétní podobu a připojit k daným proměnným. Teprve poté můžeme prvky použít na Faceplatu. Ukážeme si například prvek, který má na Faceplate zobrazovat stav hardwarového signálu. Na obrázku 32 můžeme vidět Faceplate pro kontrolní modul DIS.



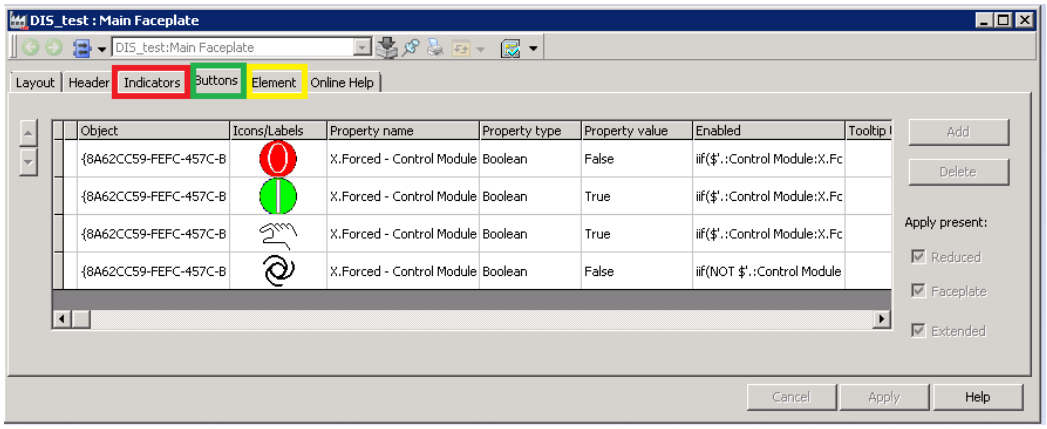
Obrázek 32 Faceplate pro prvek DIS v automatickém a manuálním módu

Na levém obrázku lze vidět, že prvek je v automatickém módu, to nám značí spirála v horní liště. Rovněž značí, že prvek je ve stavu vypnuto. Na pravém obrázku je modul v manuálním módu – obrázek ruky v horní liště a je ve stavu zapnuto. V kolonce Signal se zobrazí žlutý nápis „Forced“, pokud určíme, že chceme zadávat hodnoty ručně. V kolonce Alarm se zobrazí, zda máme alarm, alarm i událost nebo nic.

Samotný vzhled Faceplatu je řešen v konfiguračním okně Faceplatu. Zde si můžeme nastavit indikátory, tlačítka, velikost, přidat faceplate elementy atd. V záložce Indikátory (Indicators) jsou indikace zobrazující se v horní části Faceplatu, v záložce Tlačítka (Buttons) nastavujeme tlačítka a jejich funkčnost, která se zobrazí ve spodní části Faceplatu a v záložce Elementy (Elements) přidáváme námi vytvořené elementy, jako je indikace stavu a ručního zadávání hodnot, ty se poté zobrazí v prostředí části Faceplatu. Obrázky 33 a 34 graficky znázorňují jednotlivé záložky v konfiguračním okně a na Faceplatu.



Obrázek 33 Jednotlivé oblasti Faceplate



Obrázek 34 Konfigurační okno pro programování oblastí Faceplate

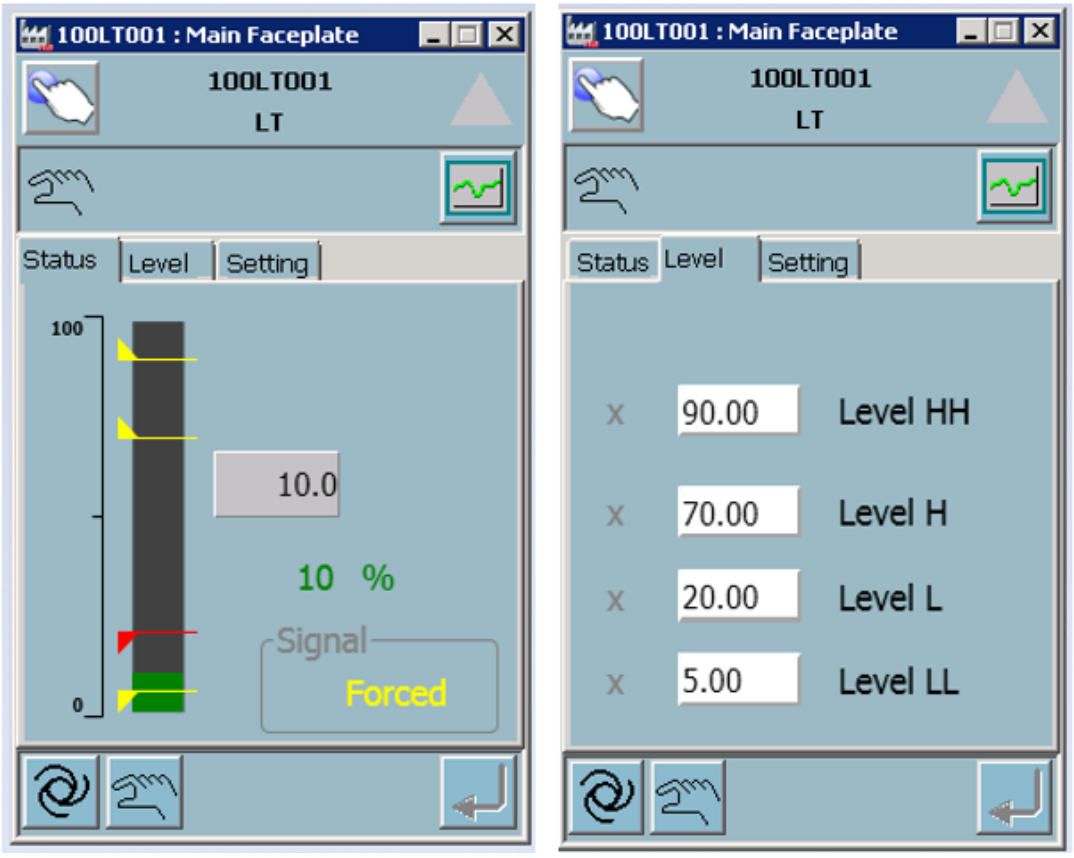
Postup vytváření modulu, psaní kódu a vytváření Faceplate se u jednotlivých modulů příliš neliší, je jen potřeba si uvědomit, jaké proměnné budeme potřebovat a co má Faceplate zobrazovat. Proto nebudu rozepisovat u následujících modulů celé postupy, jen obecný popis a hotový Faceplate.

4.2.2 Analogový vstupní signál (AIS)

Vytvoření analogového vstupního signálu bude podobné jako u DIS. Rozdíl je hlavně v alarmech. U DIS jsme měli alarm a událost pro jednu výstupní hodnotu, kdežto u AIS jsou alarmy pro 4 úrovně: HH (high-high), H (high), L (low), LL (low-low). Tato změna se projeví i na Faceplatu, kde se musí vytvořit zarážky pro určení úrovní limitů a karta pro zadávání úrovně limitů ručně.

Pro AIS byl také vytvořen Faceplate (Obrázek 35). Na záložce Status byla vytvořena indikace pomocí objektu sloupce vytvořeného v Graphic Builderu, dále byly vytvořeny zarážky pro signalizaci zadaných úrovní, vedle sloupce je rámeček, kde je zobrazována aktuální hodnota s měřenými jednotkami. Při ručním zadávání hodnot se zobrazí šedý čtvereček pro zadání požadované hodnoty. Na záložce Hladina (Level) jsou okýnka, do kterých můžeme vepisovat požadované úrovně pro zaznamenávání alarmů.

Na Faceplatu je název 100LT001 (LT = level transmitter = snímač hladiny), protože se jedná o instanci modulu AIS použitou v simulaci HW, která bude zobrazovat a řídit výšku hladiny v nádrži.

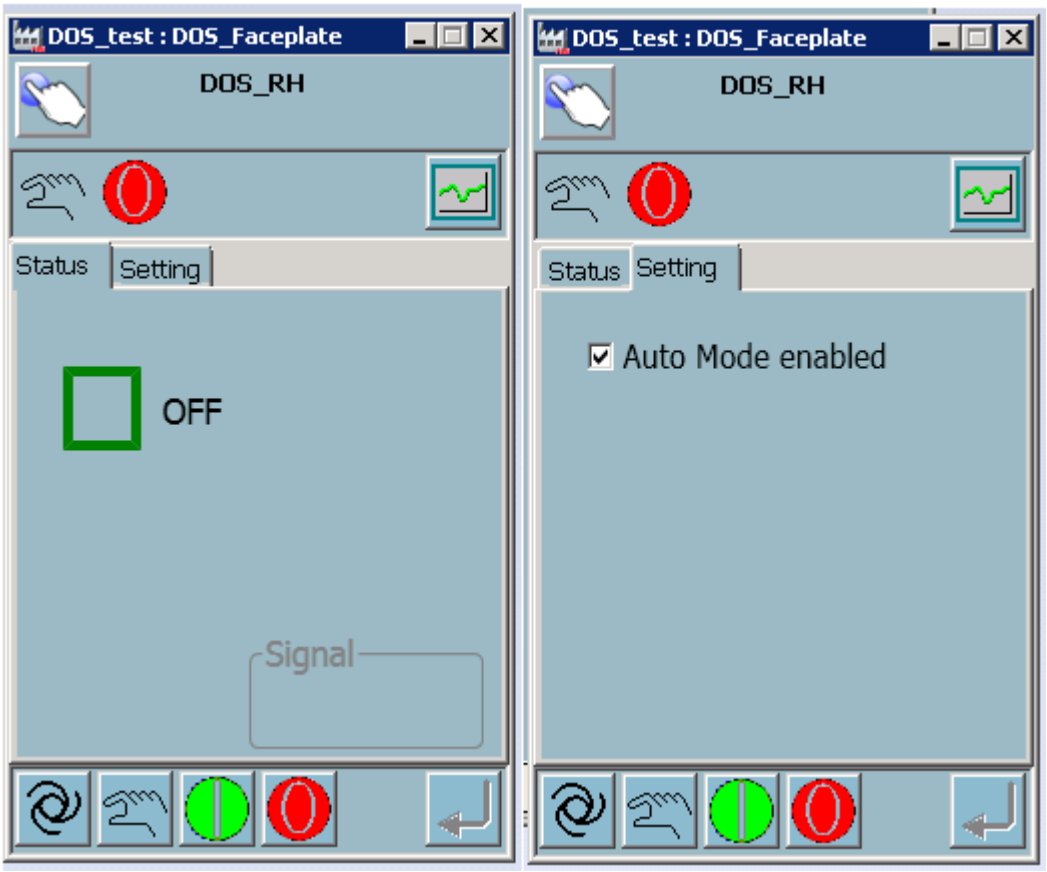


Obrázek 35 Ukázka Faceplate pro kontrolu hladiny tekutiny v nádrži a nastavení mezních hodnot alarmů

4.2.3 Digitální výstupní signál (DOS)

Kontrolní modul DOS není tak obsáhlý jako dva předešlé. V tomto kontrolním modulu nemáme alarmy ani události.

Na obrázku 36 můžeme vidět Faceplate pro kontrolní modul DOS. V záložce Status je zobrazen čtvereček indikující stav modulu a rámeček pro indikaci „Forced“, tedy ručního zadávání hodnot. Na záložce Nastavení (Settings) je umístěno tlačítko pro povolení automatického módu.

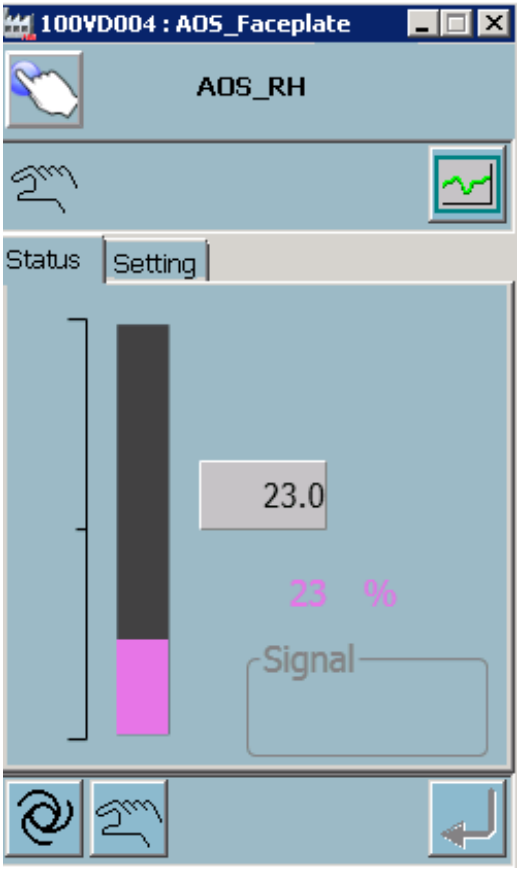


Obrázek 36 Faceplate pro DOS s ukázkou nastavení pro automatický mód

4.2.4 Analog output signal (AOS)

AOS je podobný vzhledem DOS, ale obsahuje 4 úrovně alarmů jako modul AIS. Také u tohoto modulu je potřeba si vše v editoru nadefinovat, vytvořit kód a následně Faceplate.

Na obrázku 37 je vidět Faceplate pro modul AOS. Barva sloupce je fialová, jelikož se jedná o standardní barvu pro výstupní veličinu. Na obrázku je modul v manuálním módu, a proto je zobrazen vedle sloupce šedý čtvereček pro ruční zadávání hodnot. Při přepnutí do automatického módu tento čtvereček zmizí.

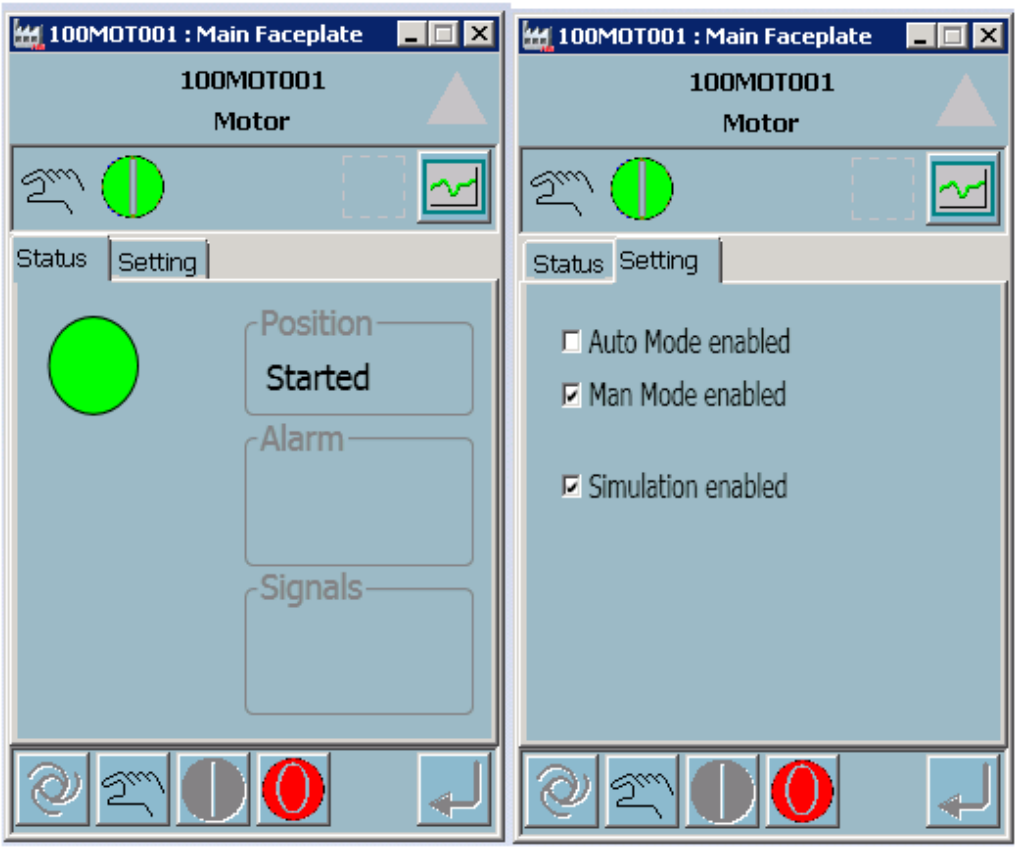


Obrázek 37 Faceplate pro modul AOS

4.2.5 Motor

Modul Motor je typu Start/Stop. Motor pracuje na principu, kdy pomocí tlačítka Start/Stop vyšleme signál, aby se motor zapnul/vypnul, a čekáme, zda se vrátí zpětná vazba o tom, že se motor zapnul. Pokud zpětná vazba přijde, tak se motor zapne a indikace změní barvu na zelenou. Jestliže zpětná vazba nepřijde, zobrazí se alarm o tom, že zpětná vazba nepřišla a indikace motoru se změní na červenou.

Graficky se značí kolečkem, které nám bude barevně zobrazovat stavy motoru: Start - zelená, Starting/Stopping – šedá (poslán signál a čeká se na zpětnou vazbu) a Stop - červená. Na obrázku 38 je zobrazen Faceplate pro motor. Na záložce Status vidíme kolečko indikující stav motoru a rámečky. Rámeček Pozice (Position) zobrazuje 3 stavy: Start, Starting/Stopping a Stop, rámeček Alarm zobrazuje alarmy, když se nevrátí zpětná vazba a rámeček Signals zobrazuje slovo „Forced“, pokud zadáváme hodnoty ručně. Na záložce Nastavení (Settings) máme 3 tlačítka: Auto mode enabled – povoluje automatický mód, Man mode enabled - povoluje manuální mód a Simulation enabled, které povoluje simulaci motoru.

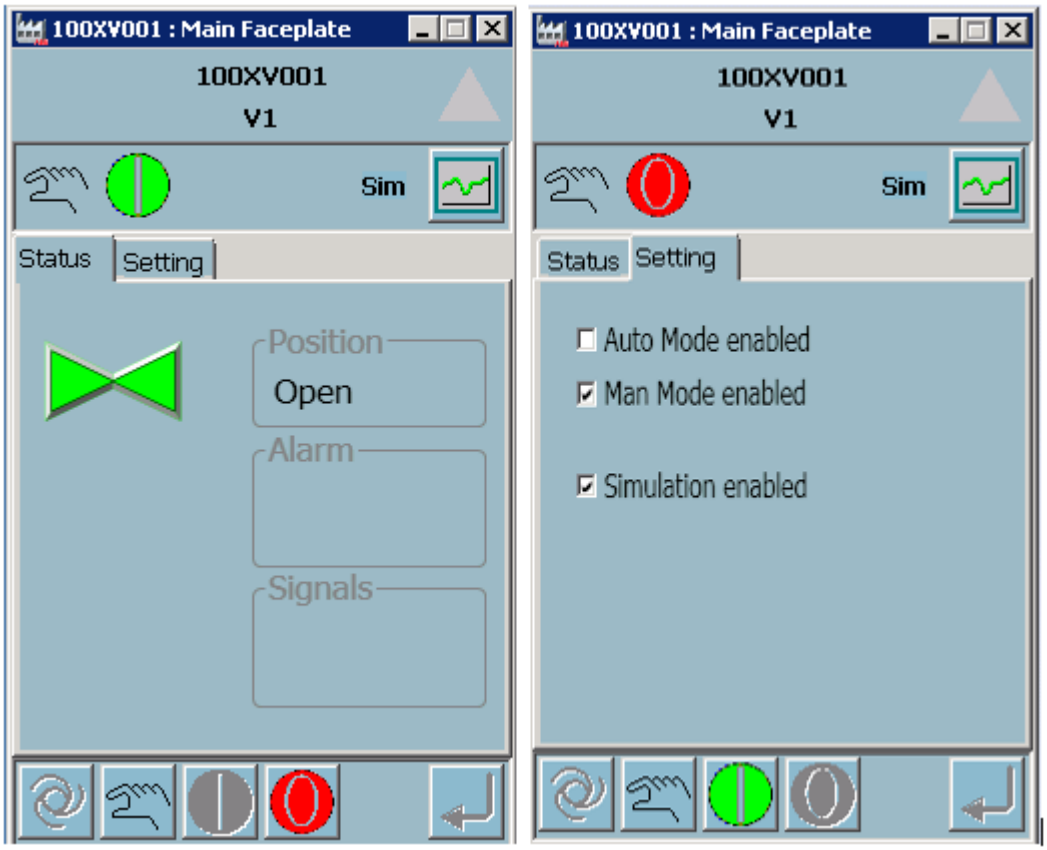


Obrázek 38 Faceplate pro modul Motor se zobrazenou záložkou Setting

4.2.6 Ventil (Valve)

Modul Ventil je podobný modulu Motor, rozdíl je v tom, že nemá stavy Start/Stop, ale Open/Close. Na Faceplatu budeme zobrazovat ventil dle standardního značení ventilu, které bude měnit barvy podle stavu. Tento stav se bude rovněž zobrazovat v rámečku – Pozice (Position) na záložce Status. Stavy jsou Open – zobrazuje se zelená barva, Travelling – šedá barva a Close – červená barva. U ventilu rovněž pošleme signál, že ho chceme otevřít/zavřít a čekáme na zpětnou vazbu. Toto čekání indikuje stav Travelling. Na záložce Nastavení (Settings) jsou 3 tlačítka se stejnými názvy i funkcemi jako u modulu Motor.

Faceplate pro ventil je zobrazen na obrázku 39. Název je změněn dle zavedeného pojmenování, tedy 100XV001, což znamená, že se jedná o Faceplate pro ventil číslo 1 (V1) z celkového počtu 6 ventilů. Zbylých 5 ventilů má shodný Faceplate, jako je na obrázku 39.



Obrázek 39 Faceplate pro modul ventil V1

5 Vizualizace hardwarového simulátoru

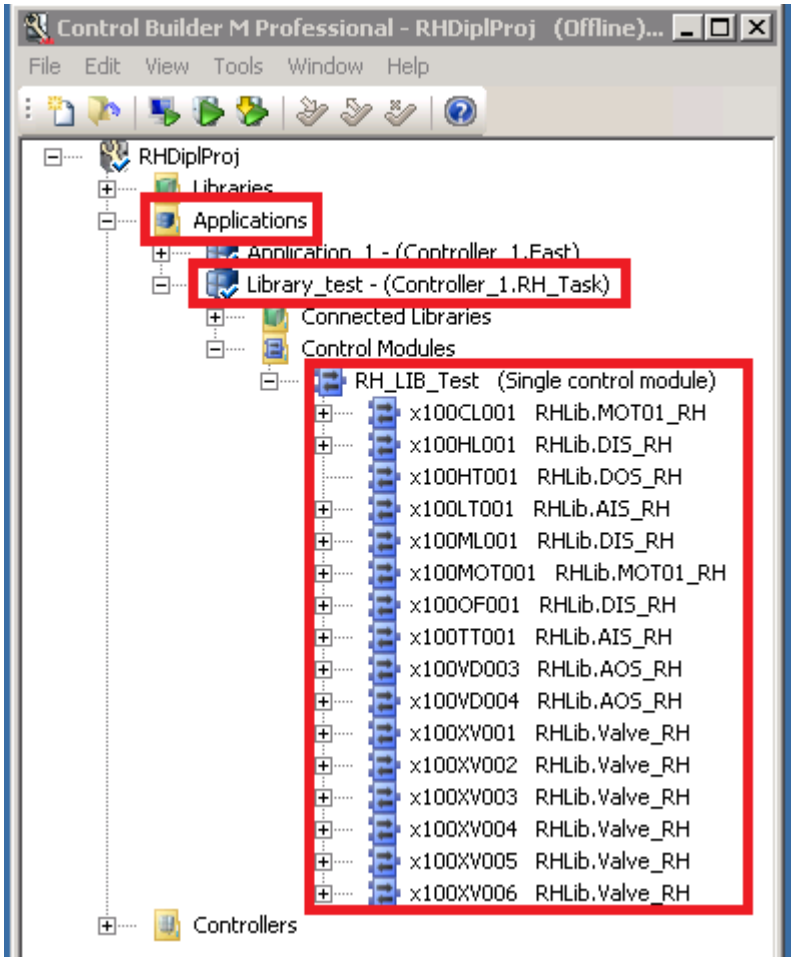
Vizualizace pomáhá operátorům a inženýrům sledovat a řídit daný technologický proces. Díky vizualizaci vidíme stav jednotlivých komponent i vzniklé chyby, které je potřeba řešit, aniž bychom reálně u dané komponenty stáli. Můžeme vzdáleně ovládat jednotlivé komponenty a tím celý proces. Je několik možností, jak ovládat a sledovat stav všech částí procesu. Na pracovišti je operátorský panel, který je procesu nejblíže, dále může být grafický displej na vzdáleném PC, celkově může být několik operátorských stanovišť. Aby se zabránilo vzájemnému překrývání a negativnímu ovlivňování procesu, jsou u každého pracoviště nastaveny parametry, které ovládání má přednost a nainstalované licence pro přístup různých lidí. To znamená, že pro operátora bude otevřeno okno s méně funkcemi a tlačítky než bude mít okno pro inženýra. Rovněž se může lišit celkový vzhled ovládání na vzdáleném PC a operátorském panelu na pracovišti. Tuto odlišnost si vytvoříme. Budeme mít grafický displej zobrazovaný na PC, kde bude více možností ovládat a kontrolovat jednotlivé komponenty. A jako druhý bude vytvořen operátorský panel PP835, na kterém bude zobrazován stav komponent, ale nebudou se dát ovládat a měnit. Budou zde 3 tlačítka, kterými se bude dát ovládat celý proces.

5.1 Simulace technologického procesu

Vytvoříme si simulaci technologického procesu napouštění a vypouštění nádrže. Tuto simulaci vytváříme pro kontrolu funkčnosti jednotlivých modulů a pro demonstraci celého procesu. V rámci simulace si můžeme vyzkoušet, jak jednotlivé části a moduly budou reagovat na naše příkazy a řízení.

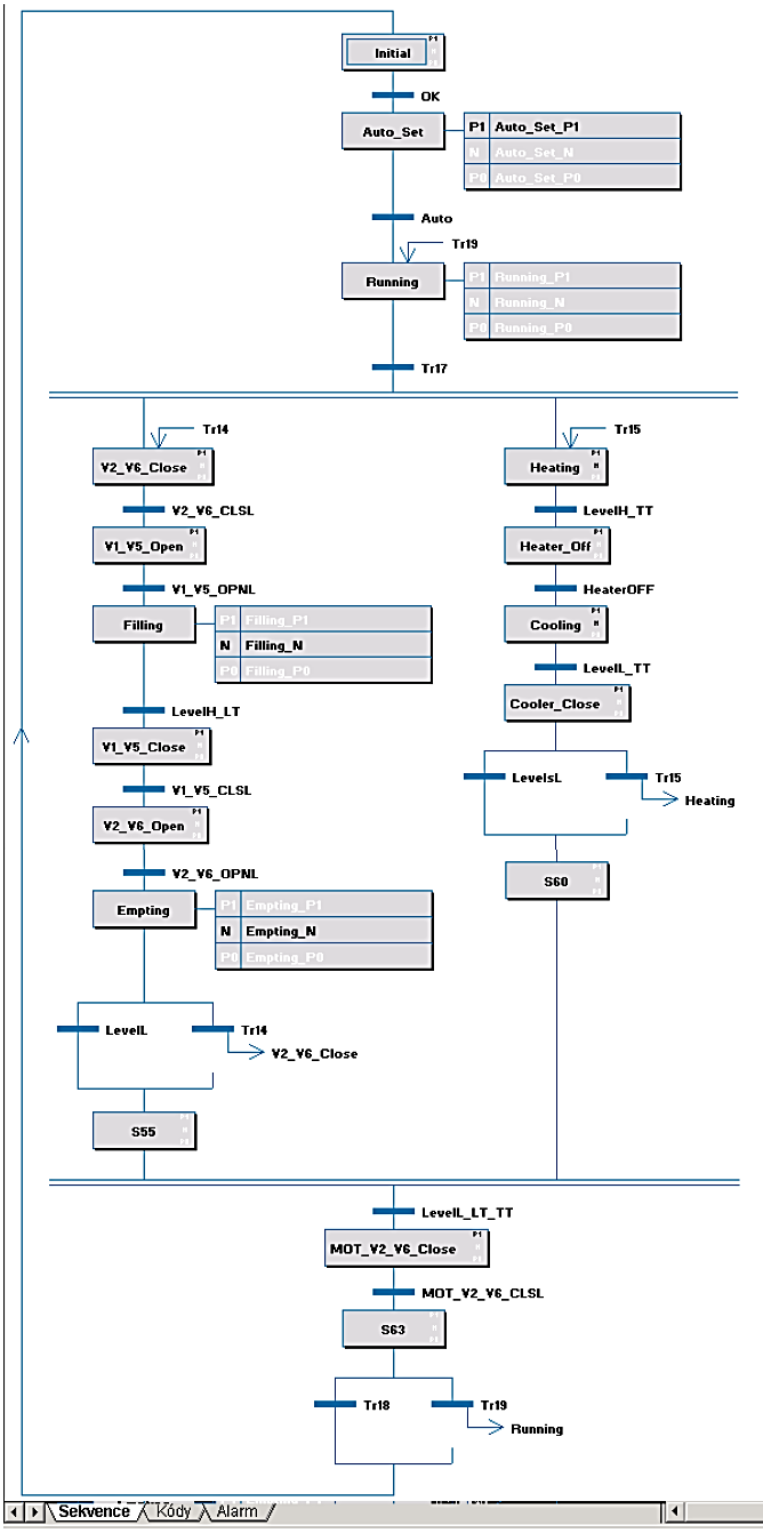
Pro simulaci vytvoříme aplikaci Library_test a v ní modul RH_LIB_Test. K tomuto modulu připojíme potřebný počet kontrolních modulů tím, že vytvoříme potřebné instance jednotlivých typů. Takže si vytvoříme instanci typu DIS a nazveme ji x100OF001, tato instance bude představovat modul pro kontrolu přetečení nádrže (Overflow), ze stejného typu uděláme ještě dvě instance a to pro signalizaci dosažení minimální (x100ML001) a maximální (x100HL001) hladiny tekutiny v nádrži. Dále vytvoříme pomocí typu DOS modul pro signalizaci stavu Ohřívače (Heater), tedy x100HT001. Typ modulu AIS nám poslouží k vytvoření modulů pro zobrazování výšky hladiny tekutiny (x100LT001) a pro zobrazování teploty tekutiny (x100TT001). Instance typu AOS budou použity pro zobrazování hodnoty otevření ventilů V3

(x100VD003) a V4 (x100VD004). Dále vytvoříme 6 instancí modulu Ventil pro ventily V1 (x100XV001), V2 (x100XV002), V3 (x100XV003), V4 (x100XV004), V5 (x100XV005) a V6 (x100XV006). A poslední typ Motor využijeme pro vytvoření 2 instancí, jedna bude pro míchadlo (x100MOT001) a druhá pro ventilátor (Cooler), tedy x100CL001.



Obrázek 40 Vytvoření aplikace a instancí jednotlivých modulů

Nyní musíme vytvořit kód pro samotnou sekvenci. K tomu si otevřeme Editor modulu RH_LIB_Test a použijeme programovací jazyk SFC (Sequential Function Chart). Tímto jazykem vytvoříme sekvenci pro celý technologický proces (Obrázek 41).



Obrázek 41 Sekvence pro simulaci hladiny a teploty v nádrži

Na Obrázku 41 můžeme vidět vzhled výsledné sekvence. Ta probíhá v několika krocích. Prvním krokem je Initial, kde se všechny komponenty zastaví, popřípadě zavřou. Tento krok je důležitý pro ovládací panel PP835, což vysvětlím v kapitole 5.3. Následuje podmínka OK, v této podmínce je příkaz k zapnutí simulace. V kroku Auto_Set se nastaví všechny komponenty do automatického módu, aby mohla samostatně probíhat

simulace. Podmínka Auto kontroluje, zda se všechny části opravdu přepnuly do automatického módu. Krok Running je zde proto, aby sekvence probíhala stále dokola.

Nyní se dostáváme do paralelní sekvence, kdy ve stejnou chvíli probíhají dvě simulace. V levém sloupci probíhá simulace napouštění a vypouštění nádrže. A v pravém sloupci simulace ohřívání a ochlazování tekutiny. Nyní oba sloupce blíže vysvětlím.

Začnu levým sloupcem, tedy simulací napouštění a vypouštění nádrže. Tato sekvence začíná krokem, kdy se ventily V2 a V6 zavřou, následně se ventily V1 a V5 otevřou a začne napouštění. Pro napouštění je v záložce Kódy vytvořen kód, který počítá, jakou rychlostí se má nádrž napouštět. Výpočet je založen na principu, kdy odečteme výstupní průtok od vstupního, a vyjde nám hodnota, o kterou se bude hladina zvedat/snižovat. Vstupní průtok je vypočítán pomocí výstupních hodnot ventilů V1, V5 a V3. A výstupní průtok pomocí výstupních hodnot ventilů V2, V4 a V6. Výpočet je vidět na obrázku 42. Po napuštění nádrže po hranici námi určenou se aktivuje alarm a zavřou se ventily V1 a V5, následně se otevřou ventily V2 a V6 a nádrž se začne vypouštět. Vypouštění pokračuje, dokud nebude výška hladiny na nejnižší přípustné hladině, kterou jsme si určili. Poté začne cyklus znova.

```
if vSimulationON then
vSimFlowIN:= bool_to_real(sig100XV001_Y.Value) * (sig100VD003_Y.Value/200 + 0.5*bool_to_real( sig100XV005_Y.Value));
vSimFlowOUT:= bool_to_real(sig100XV002_Y.Value) * (sig100VD004_Y.Value/200 + 0.5*bool_to_real( sig100XV006_Y.Value));
vSimFlow:=vSimFlowIN-vSimFlowOUT;

if vSimLevel>=0 and vSimLevel<=100 then
   vSimLevel:=vSimLevel+vSimFlow;
elsif vSimLevel< 0 then
   vSimLevel:=0;
elsif vSimLevel > 100 then
   vSimLevel:=100;
end_if;

sig100LT001_X.IOValue:=vSimLevel;
```

Obrázek 42 Kód pro výpočet přírůstku hladiny v nádrži

V pravém sloupci máme simulaci ohřívání a ochlazování tekutiny v nádrži. Sekvence začíná tím, že zapneme ohřívač a ten ohřívá tekutiny do hodnoty, kterou jsme si nastavili. Poté se ohřívač vypne a zapne se ventilátor, ten bude ochlazovat po minimální přípustnou hodnotu, kterou jsme si nastavili. Celý cyklus se stále opakuje, přičemž pro ohřívání a ochlazování byl rovněž vytvořen kód v záložce Kódy (Obrázek 43).

```
(** Heating/Cooling **)

vSimHeating := (bool_to_real(sig100HT001_Y.Value)) -  (bool_to_real(sig100CL001_Y.Value));

if vSimTemp>=0 and vSimTemp<=100 then
   vSimTemp := vSimTemp + vSimHeating * 0.5;

elsif vSimTemp< 0 then
   vSimTemp:=0;
elsif vSimTemp > 100 then
   vSimTemp:=100;
end_if;

sig100TT001_X.IOValue := vSimTemp;
```

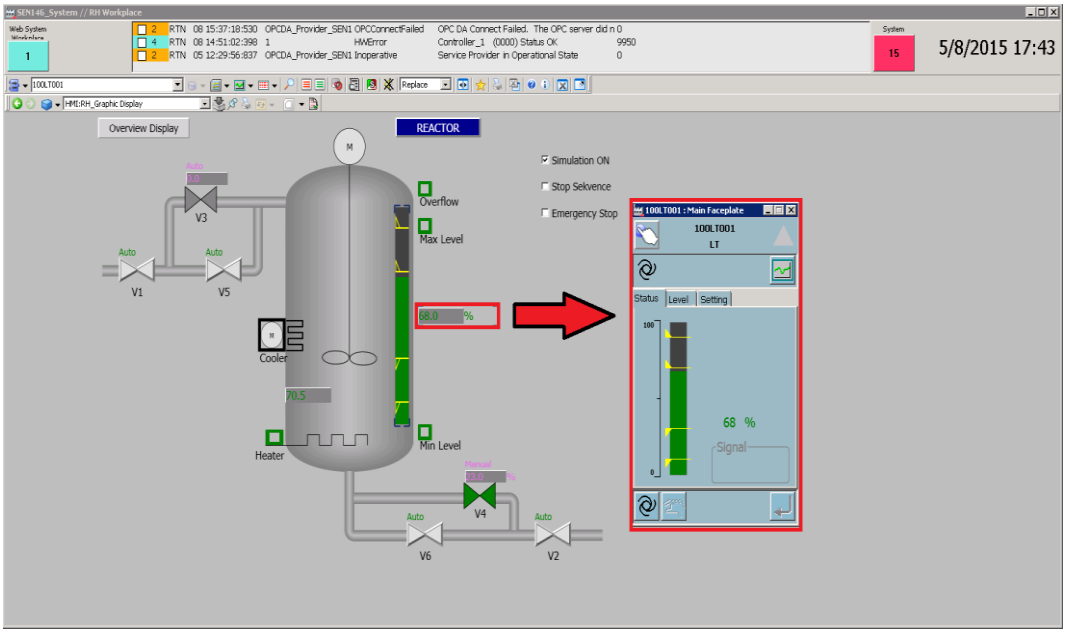
Obrázek 43 Kód pro výpočet teploty v nádrži

V průběhu obou simulací se otáčí mixér a promíchává tekutinu tak, aby byla ohřátá (ochlazena) rovnoměrně.

5.2 Grafický displej na PC

Pro ovládání a sledování procesu z počítače vytvoříme grafický displej, na kterém bude vidět celý průběh. Na vytvoření grafického displeje použijeme programovací prostředí Graphic Builder. Zde pomocí předdefinovaných tvarů vytvoříme nádrž a potrubí. K této nádrži přidáme grafické elementy našich komponent. To znamená, že přidáme 6 ventilů, 2 displeje zobrazující procentuálně hodnotu otevření ventilů V3 a V4, motor mixéru, sloupec pro zobrazení výšky hladiny tekutiny a k němu displej zobrazující hodnotu naplnění nádrže. K úplnosti sledování výšky hladiny přidáme 3 čtverečky indikující přetečení, maximální a minimální hladinu. Dále přidáme zobrazení ohřívače a ventilátoru a displej pro zobrazení teploty tekutiny v nádrži. A naposled přidáme tlačítko pro zapnutí simulace. Tento grafický displej nám totiž poslouží pro zobrazení probíhající simulace i pro zobrazení stavu reálného hardwaru. Záleží na našem nastavení.

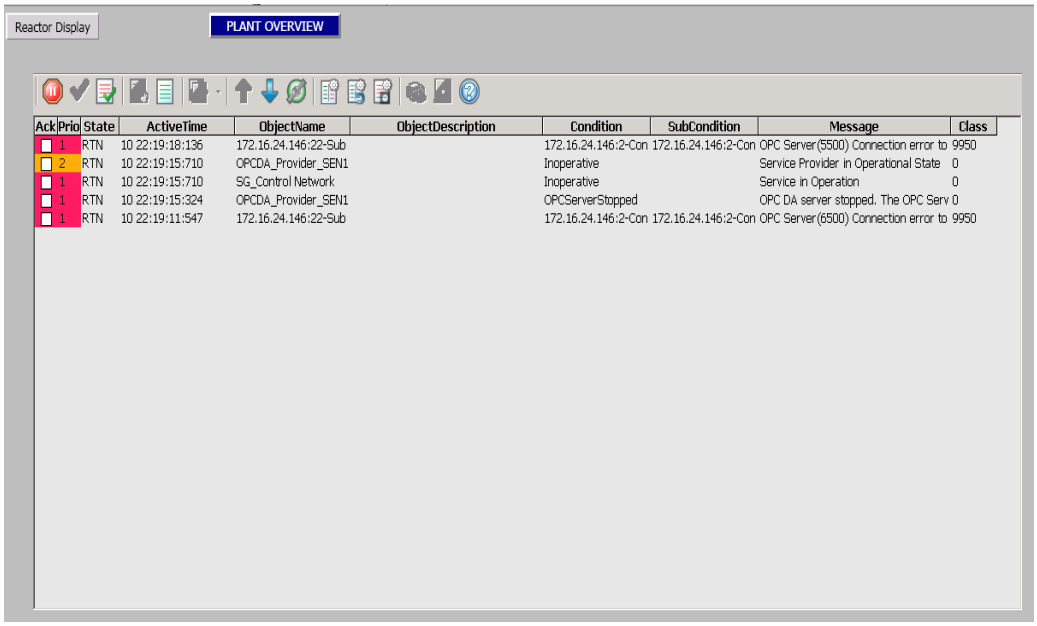
Při kliknutí na jakoukoliv komponentu (ventil, ohřívač, atd.) se nám zobrazí odpovídající Faceplate, kterým můžeme danou komponentu ovládat. Celý grafický displej i s ukázkou Faceplate je zobrazen na Obrázku 44.



Obrázek 44 Ukázka otevření Faceplate v grafickém displeji

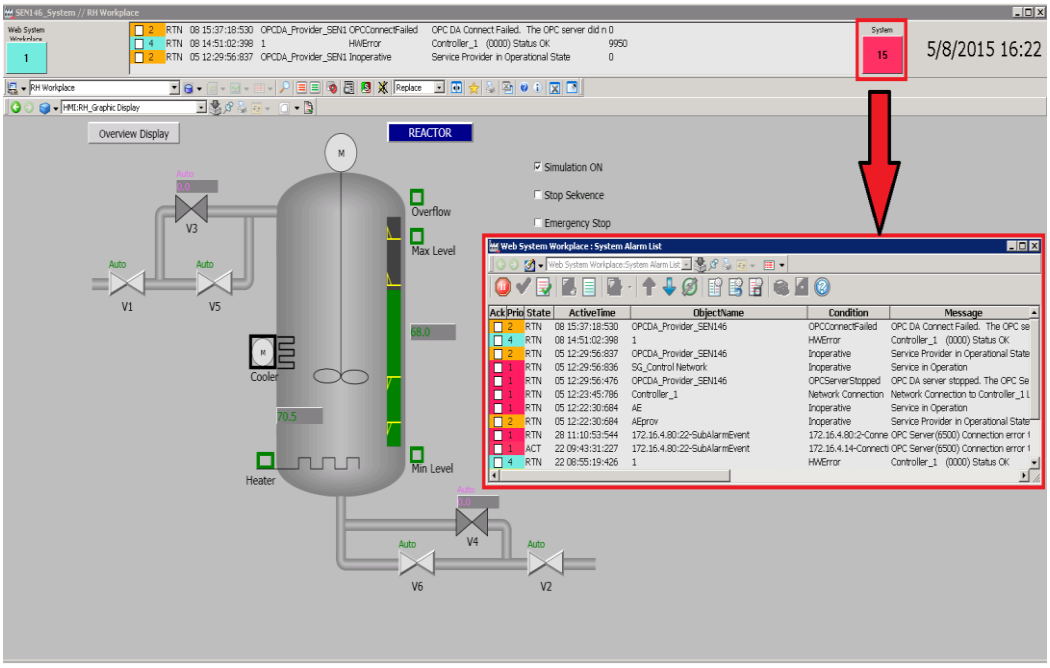
Na obrázku 44 lze vidět prostředí pro operátora, na kterém je tlačítko Overview Display, kterým se dostaneme do nového okna, ve kterém jsou zobrazeny všechny alarmy,

případně události k jednotlivým komponentům (Obrázek 45). Tlačítkem Reactor Display se opět vrátíme do okna s nádrží.



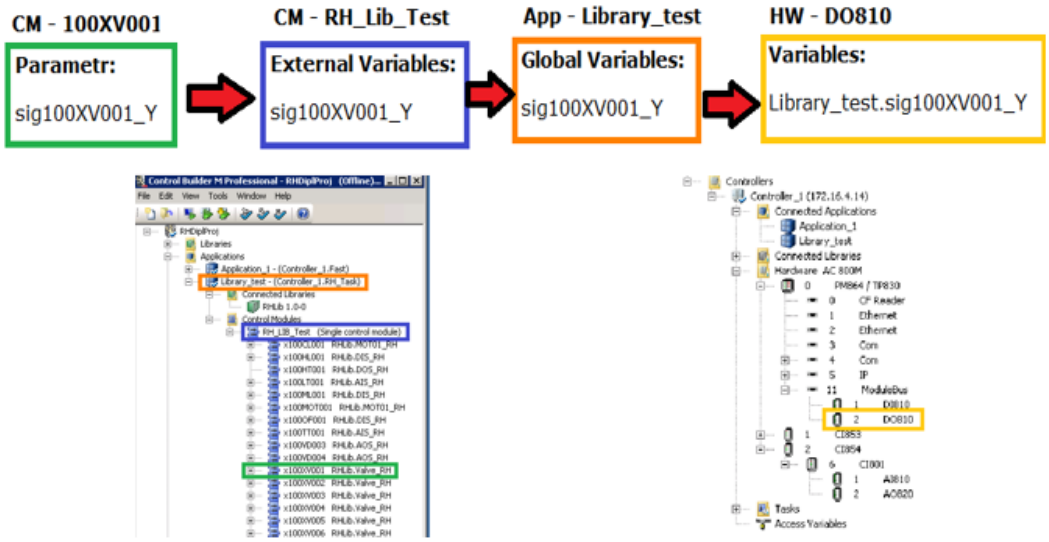
Obrázek 45 Overview displej se seznamem alarmů

V horní části grafického displeje je vidět lišta, která zobrazuje aktuální alarmy. Kompletní seznam alarmů můžeme otevřít tím, že klikneme na rámeček vedle lišty (Obrázek 46).



Obrázek 46 Alarmy na Grafickém displeji

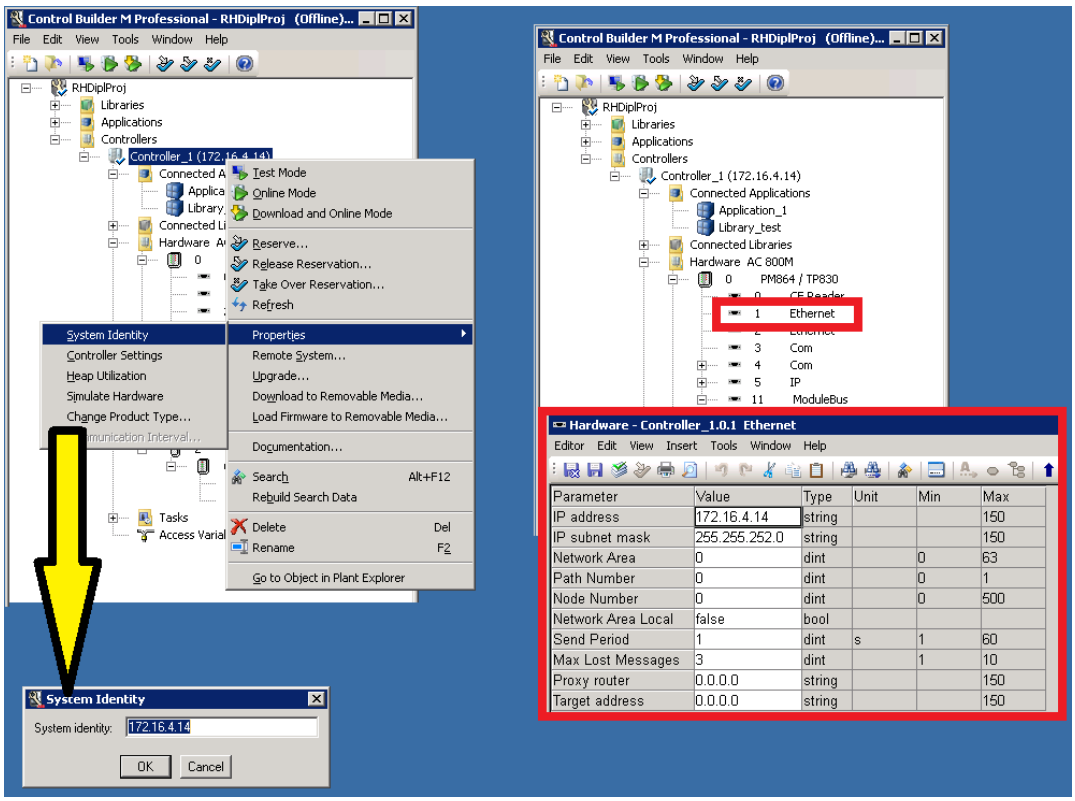
Aby simulace mohla probíhat, musí být správně propojeny signály. Ukážeme si ukázku připojení signálu Y z modulu x100XV001 (Obrázek 47). Pravým kliknutím myši na modul x100XV001 si otevřeme okno Connections. Zde si k proměnné Y přiřadíme signál sig100XV001_Y. Tento signál si nadefinujeme v kontrolním modulu RH_LIB_Test v záložce External Variables. A poté si signál se stejným názvem nadefinujeme i v samotné aplikaci Library_test, kde ho vepíšeme do sloupce Global Variables.



Obrázek 47 Ukázka propojení signálu z kontrolního modulu až do aplikace

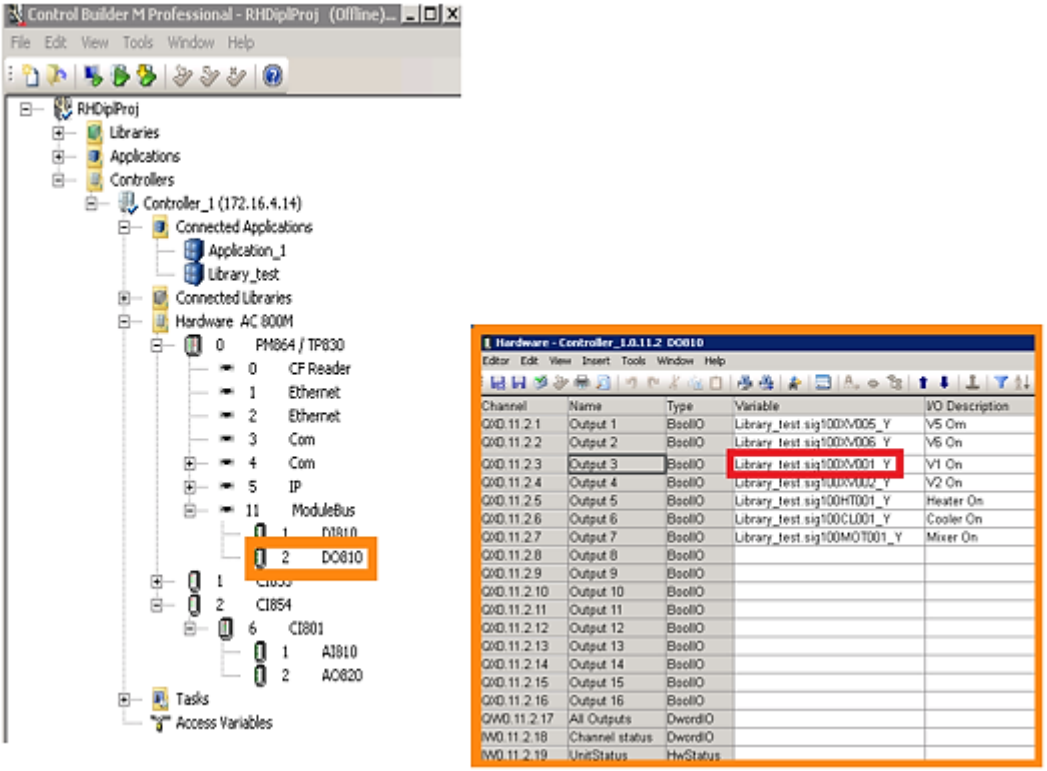
Jednotlivé proměnné jsou napojeny na grafické elementy daných modulů a proto se nám v tuto chvíli, již bude na grafickém displeji zobrazovat průběh simulace.

Pro zobrazení simulace na grafickém displeji použijeme tlačítko Simulation na displeji s nádrží. Simulace může probíhat na simulovaném kontroléru (SoftController) nebo na reálném (AC800M). Pro nastavení požadovaného kontroléru musíme v programu Control Builder nastavit v záložce Controllers IP adresu daného kontroléru. Pro SoftController je to adresa: 172.16.4.80:2 (stejná jako adresa počítače s přidáním: :2), a pro reálný kontrolér je IP adresa 172.16.4.14. Vybranou IP adresu nastavíme přímo pro kontroler v Control Builderu a poté v záložce Hardware -> Ethernet ->Settings.



Obrázek 48 Nastavení IP adresy Kontroleru

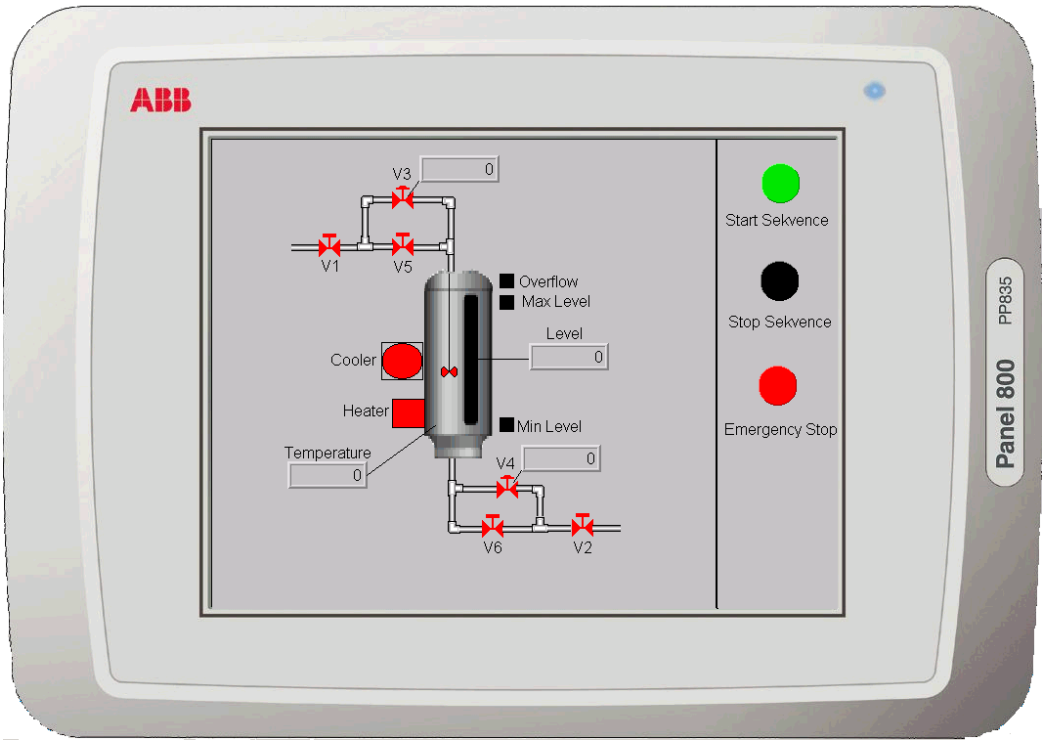
Pro sledování procesu z reálného kontroléru nastavíme nejprve výše uvedenou IP adresu a poté musíme vytvořit strukturu reálného HW v prostředí Control Builderu. To znamená, že v záložce Controllers -> Hardware AC800M přidáme procesorovou jednotku PM864/TP830 a následně v záložce ModuleBus přidáme jednotlivé I/O karty, které jsou na kontrolér připojeny. V našem případě se jedná o karty AI810, AO820, DI810 a DO810. Aby se nám zobrazovaly na grafickém displeji hodnoty z reálného hardwaru, musíme mít správně připojeny signály na jednotlivé I/O karty. Pomocí manuálu zjistíme, na který port dané karty máme signál připojit, aby plnil správnou funkci. Dále musíme signály, které máme nadefinované v záložce Globální proměnné (Global Variables) v Editoru aplikace Library_test připojit na karty pomocí sloupce Proměnná (Variable) u každé karty. Na obrázku 49 je vidět propojení signálu sig100XV001_Y z aplikace na kartu DO810.



Obrázek 49 Ukázka připojení signálu na hardwarovou kartu

5.3 Operátorský panel PP835

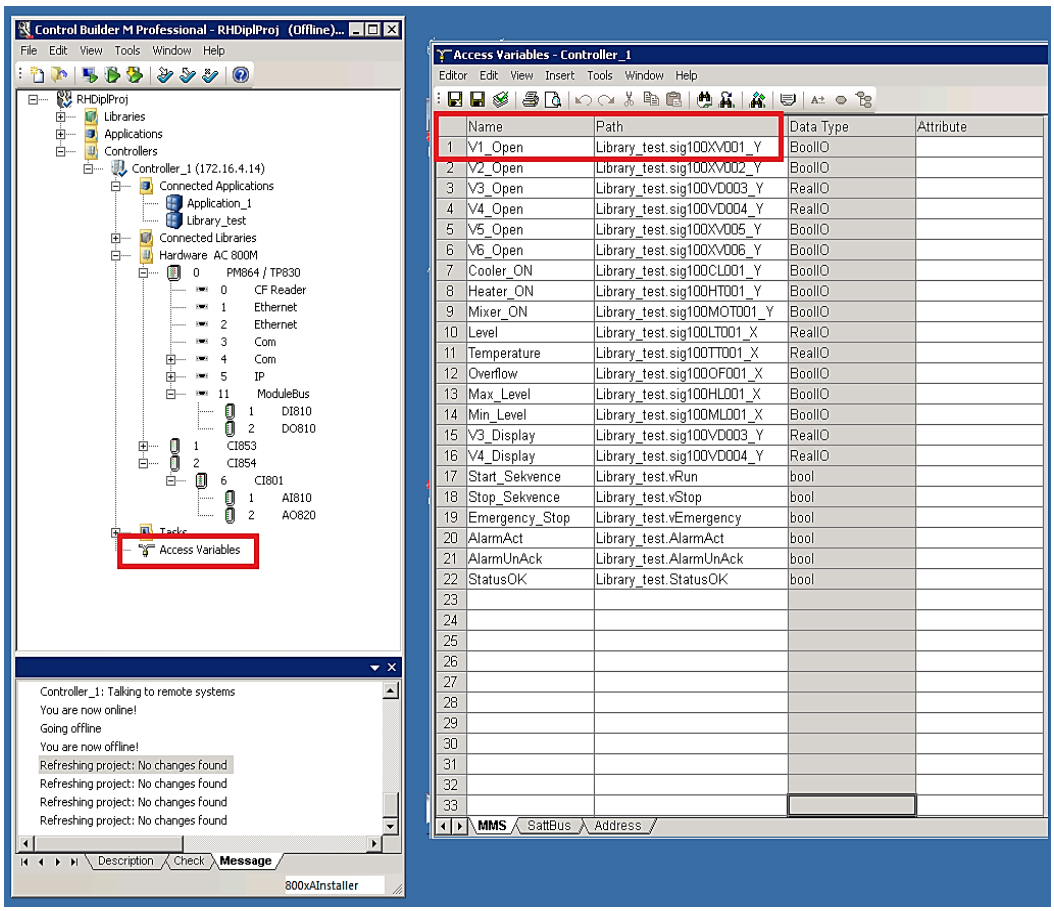
Operátorský panel PP835 je dotykový ovládací panel určený pro sledování průběhu technologického procesu přímo na pracovišti. V rámci našeho projektu jsme vytvořili v programu Panel Builder 800 panel, který je rozdělen do dvou částí (Obrázek 50).



Obrázek 50 Operátorský panel PP835

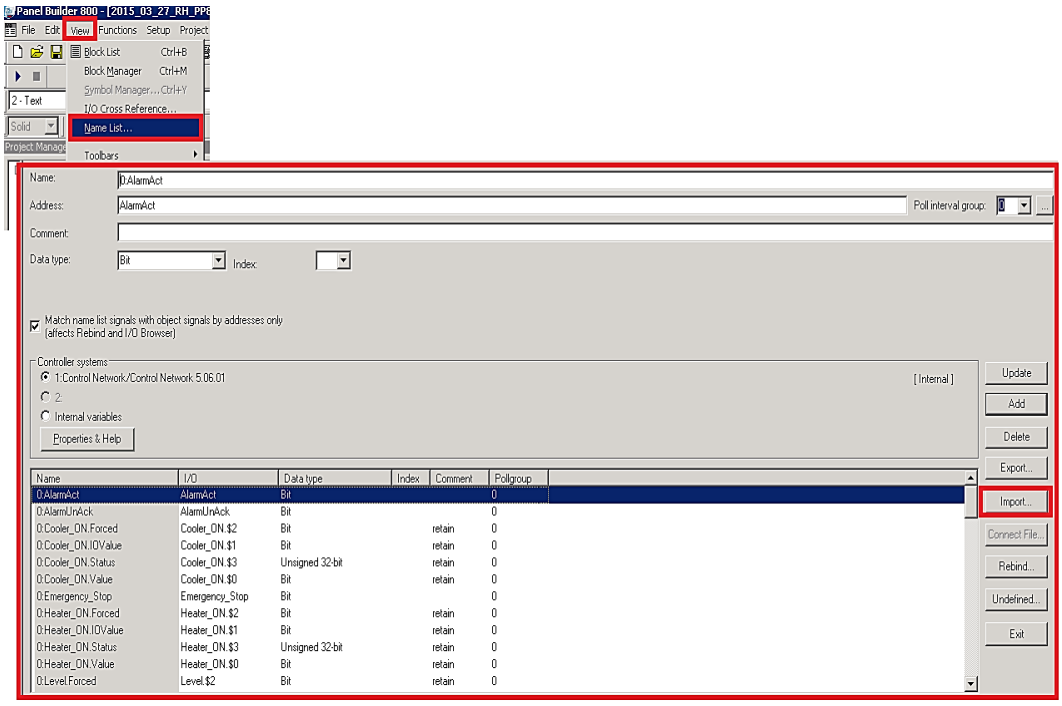
V levé části jsme vytvořili obrázek nádrže se všemi použitými komponenty, jako na grafickém displeji. U každé komponenty jsme připojili odpovídající proměnnou, na základě které bude komponenta měnit barvu.

Propojení proměnných si ukážeme opět na ventilu V1. V programu Control Builder otevřeme okno Access Variables, kde si nadefinujeme v záložce MMS proměnnou V1_Open. Této proměnné ve sloupci Path přiřadíme cestu k proměnné sig100XV001_Y, kterou již máme nadefinovanou v aplikaci Library_test (Obrázek 51).



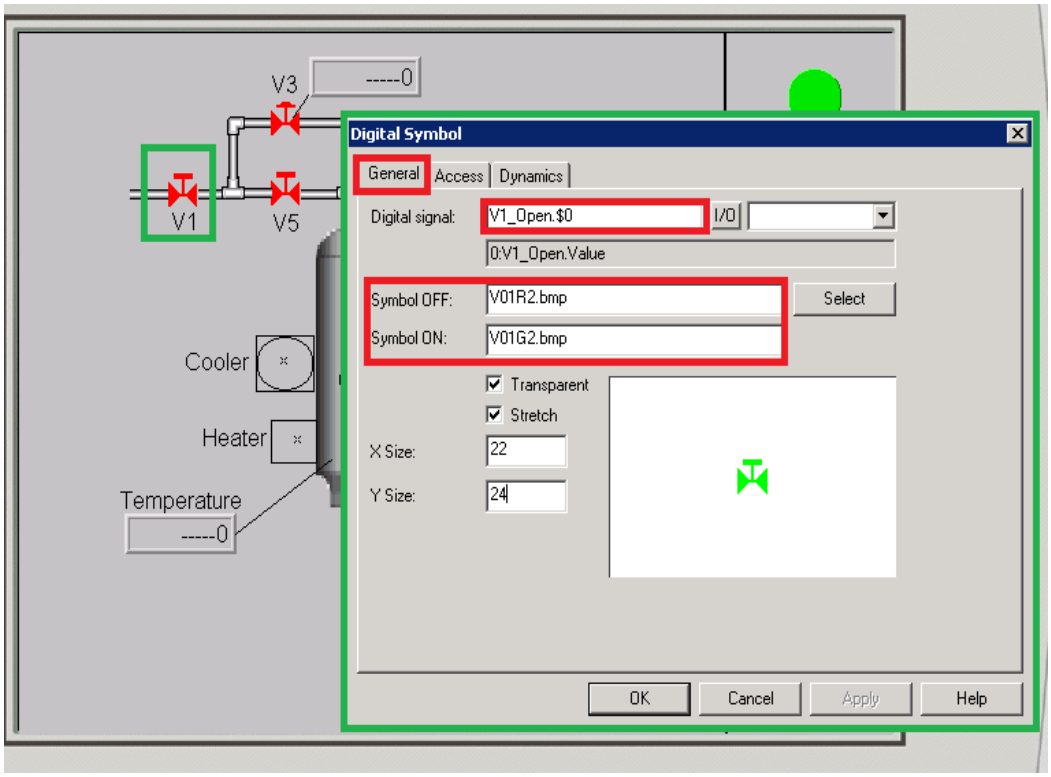
Obrázek 51 Propojení signálu s Access Variables

Nyní v programu Panel Builder 800 rozklikneme v horní liště záložku View -> Name List a naimportujeme si vytvořené Access Variables (Obrázek 52).



Obrázek 52 Importování Access Variables do Panel Builderu 800

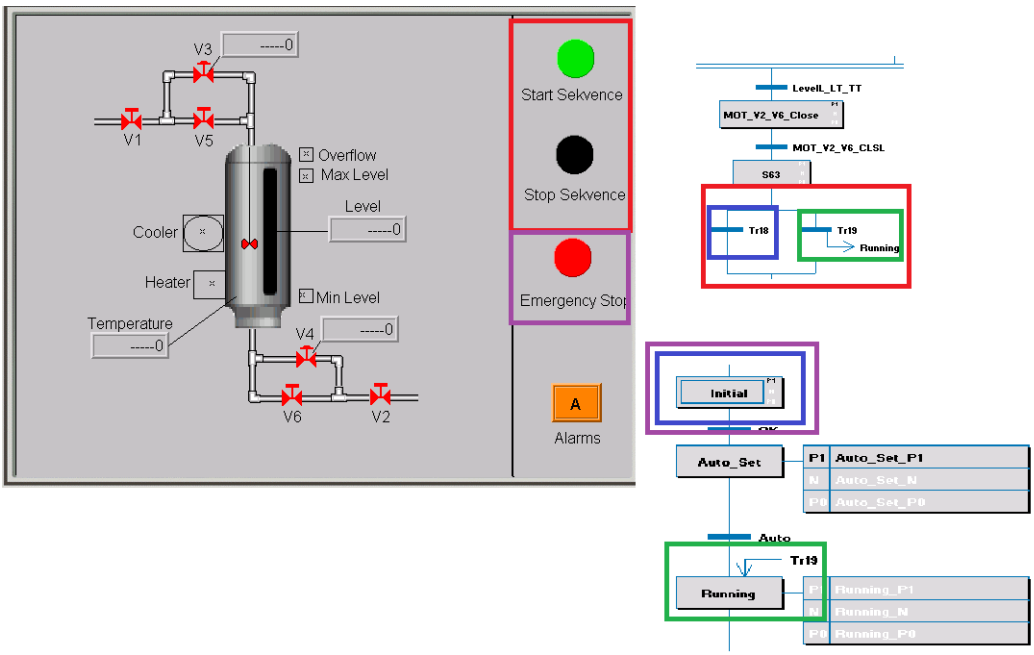
Dále si rozklikneme objekt V1, zde v záložce General přiřadíme do kolonky Digital signal námi vytvořenou proměnnou V1_Open. Pro změnu barvy vložíme do kolonky Symbol OFF, obrázek červeného ventilu a do kolonky Signal On obrázek zeleného ventilu (Obrázek 53).



Obrázek 53 Připojení proměnné k indikaci stavu ventilu V1

Takto postupujeme i u ostatních komponent. Nyní si popíšeme pravou část panelu.

V pravé části panelu se nacházejí 4 ovládací tlačítka: Start Sekvence, Stop Sekvence, Emergency Stop a Alarms. Prvními třemi tlačítky se dostáváme do aplikace a tedy sekvence. Tlačítko Start Sekvence spustí sekvenci v bodě OK a sekvence poběží, dokud nezmáčkneme tlačítko Stop Sekvence. Po zmáčknutí tlačítka Stop Sekvence doběhne celá sekvence do konce a pak se teprve zastaví. Tato tlačítka jsou napojena na proměnnou vRun z aplikace a ta je použita v sekvenci v podmínkách OK, Tr18 a Tr19. Podmínky Tr18 a Tr19 jsou rozhodující krok. V tomto bodě se rozhodne, zda se sekvence vrátí zpět do bodu Running a zopakuje se nebo zda se zastaví a vrátí se do kroku Initial. Jestliže potřebujeme sekvenci z nějakého důvodu okamžitě v jakémkoliv bodě vypnout, tak použijeme tlačítko Emergency Stop. Tímto tlačítkem se celá sekvence zruší, vše se vypne a zavře. A znovu sekvenci pustíme až tlačítkem Start Sekvence.



Obrázek 54 Propojení panelu PP835 a sekvence

Tlačítko Alarms nám umožňuje otevřít seznam všech alarmů, které v rámci sekvence vznikly.

6 Zhodnocení

Pro ověření funkčnosti navržených knihoven a řídicího programu pro hardwarový simulátor jsem nejprve simulátor připojila na simulovaný hardware (SoftController) a spustila na grafickém displeji vnitřní simulaci procesu. Vytvořená sekvence pro technologický proces napouštění a vypouštění nádrže proběhla bez problémů. Vyzkoušela jsem tedy připojit HW simulátor na reálný kontrolér ABB AC800M a vypnout vnitřní simulaci. Kontrolér a operátorské pracoviště vytvořené na PC spolu komunikovaly. Lišili se pouze v analogových hodnotách pro výšku hladiny v nádrži. Tato odlišnost je způsobena nefunkčními analogovými signály na hardwarovém simulátoru OSLO. Z tohoto důvody jsem v aplikaci pro simulaci hladiny a teploty v nádrži vytvořila kód pro jejich výpočet.

Dále jsem ověřila komunikaci mezi HW simulátorem a operátorským pracovištěm na panelu PP835. Vizualizace se shodovala se stavem jednotlivých komponent, poté jsem vyzkoušela funkčnost třech řídicích tlačítek. Tlačítkem Start Sekvence se měla celá sekvence zapnout od začátku a běžet, dokud není zmáčknuto tlačítko Stop Sekvence nebo Emergency Stop. Po zapnutí Start Sekvence, opravdu cyklus probíhal ve smyčce stále dokola. Pro vypnutí jsem použila nejprve tlačítko Stop Sekvence, které má za úkol vypnout sekvenci po ukončení cyklu. Cyklus doběhl do konce a zastavil se. Zapnout se dal opět až tlačítkem Start Sekvence. Spustila jsem cyklus znovu, abych vyzkoušela tlačítko Emergency Stop, které má sekvenci vypnout v jakémkoliv kroku a vypnout/zavřít všechny komponenty procesu. I toto tlačítko fungovalo podle zadání.

Tuto práci by bylo možné rozšířit přidáním dalších tlačítek pro ovládání, například jen pro část napouštění nebo vypouštění, pouze pro ovládání jen teploty v nádrži, pro ovládání jednotlivých ventilů z dotykového panelu atd.

7 Závěr

Cílem diplomové práce bylo navrhnout knihovnu pro řídicí systém ABB AC800M tak, aby byl použitelný pro demonstrační a školící účely firmy ABB.

V první části jsem se obeznámila s prostředím ABB 800xA, kde jsem prostudovala programovací nástroje, potřebné k vytvoření projektu. Seznámila jsem se tedy s programovacím prostředím Control Builder M Professional, Graphic Builder a Panel Builder 800. Dále s programy OPC a SoftController, důležitými pro správnou funkčnost celého projektu. Po seznámení s programy a s možnostmi jejich využití v rámci mého projektu jsem popsala vzhled a funkčnost hardwaru, který byl při mé práci použit. Jedná se o kontrolér PM864 řady AC800M, vstupní a výstupní karty AI810, AO820, DI810 a DO810. Poslední částí hardwaru je dotykový operátorský panel PP835, používaný pro ovládání procesu ze stanoviště operátora.

Před začátkem vytváření projektu bylo ještě potřeba prostudovat funkční popisy stávajících knihoven použitých v projektech firmy ABB. V těchto funkčních popisech jsem zjistila, jak vypadá struktura knihovny, jaké moduly se používají a jak se programují funkcionality jednotlivých modulů.

Po prostudování všech potřebných materiálů jsem ve druhé části své práce vytvořila základní knihovnu, která obsahovala 6 prvků: AIS (analogový vstupní signál), AOS (analogový výstupní signál), DIS (digitální vstupní signál), DOS (digitální výstupní signál), Motor a Ventil. Těmto prvkům jsem vymyslela kódy pro zajištění jejich funkčnosti, grafické elementy pro vizualizaci a faceplaty (ovládací obrazovky). Dále jsem vytvořila aplikaci, ve které jsem udělala množství instancí modulů z knihovny, které odpovídají hardwarovému simulátoru OSLO, tedy 6 ventilů, motor pro mixér, ohřívač, ventilátor, zobrazení teploty a hladiny, indikace přetečení, maximální a minimální hladiny. Pro tyto prvky jsem vytvořila pomocí sekvence řídicí aplikaci simulující proces pro napouštění a vypouštění hladiny a ohřívání a chlazení tekutiny uvnitř nádrže. Navrhla jsem pro tento proces grafický displej zobrazující průběh simulace.

Vytvořenou aplikaci jsem připojila přes signály na hardware a vyzkoušela funkčnost nejdříve přes simulovaný kontrolér (SoftController) a následně přes reálný kontrolér ABB AC800M.

Pro ovládání procesu jsem vytvořila dvě operátorské pracoviště. Jedno pro vzdálené ovládání a druhé pro lokální ovládání. Pro ovládání procesu ze vzdáleného PC byl vytvořen grafický displej obsahující vzhled procesu s ovládacími obrazovkami pro jednotlivé komponenty a seznam alarmů vzniklých v průběhu procesu. Tento grafický displej se používá na velícím stanovišti a umožňuje komplexní ovládání celého procesu.

A pro lokální ovládání procesu na pracovišti byl vytvořen Panel PP835, a v něm 3 základní tlačítka: Start Sekvence, Stop Sekvence a Emergency Stop. Navrhla jsem zde také vizualizaci celého procesu, aby se zobrazovaly stavy jednotlivých komponent. Tento panel se umisťuje blízko zařízení a umožňuje řízení procesu jako celku, může vypnout a zapnout proces, ale neovládá jednotlivé prvky jako ventil apod.

8 Použitá literatura

[1] ABB, *AC 800M Controller Hardware, System Version 5.1*, 2013.452 stran.

[2] ABB, *AC 800M and S800 I/O Getting Started,* 2009.68 stran.

[3] ABB, *Course T315C System 800xA with AC 800M Engineering, Part1 – Control Builder*, 2013.

[4] ABB, *Industrial^{IT} Panel 800 – PP835 Version 5.1, Hardware and Installation,* 2011, 54 stran.

[5] ABB, *Industrial^{IT} Panel Builder 800 Version 5.0, Programming and Installation,* 2006, 362 stran.

[6] ABB, *Panel for TestField,* 2013, 72 stran

[7] ABB, *Sevan300VMSLib2.1-0*, 32 stran

[8] ABB, *S800 I/O Modules and Termination Units, System Version 5.1*, 2013. 668 stran.

[9] ABB, *T315H System 800xA with AC 800M Engineering, Part 2 – Human System Interface*, 2013.

[10] Farana, R., Smutný, L., Víteček, A., Vítečková, M., Wagnerová, R., *Doporučení pro psaní odborných textů z oblasti automatizace a informatiky.* Ostrava: VŠB-TUO, 2008. ISBN 978-80-248-1925-9

[11] Zavadil, J. *Systém monitorování a řízení technologie. Ostrava: Katedra automatizační techniky a řízení*, VŠB-TU Ostrava, 2009. 57 stran. Diplomová práce, vedoucí: Landryová, L.

Seznam obrázků a tabulek